Engineering
Simplicity

Cloud Native Contrail Networking

Installation and Life Cycle Management Guide for Rancher RKE2

Published
2023-09-28

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

YEAR 2000 NOTICE

END USER LICENSE AGREEMENT

Table of Contents

1 **Introduction**

Cloud-Native Contrail Networking Overview | 2

Terminology | 4

CN2 Components | 6

Deployment Models | 11

- Single Cluster Deployment | **11**
- Multi-Cluster Deployment | **12**

System Requirements | 15

2 **Install**

Overview | 17

Before You Install | 18

Install Single Cluster CN2 on Rancher RKE2 | 19

- Install Single Cluster CN2 on Rancher RKE2 Running Kernel Mode Data Plane | **21**
- Install Single Cluster CN2 on Rancher RKE2 Running DPDK Data Plane | **24**

Install Multi-Cluster CN2 on Rancher RKE2 | 28

Install Contrail Tools | 29

- Install ContrailReadiness Controller | **30**

Manifests | 31

- Manifests in Release 23.3 | **31**
- Contrail Tools in Release 23.3 | **32**
- Contrail Analytics in Release 23.3 | **33**

3 **Monitor**

Overview | 36

Install Contrail Analytics and the CN2 Web UI | 36

Kubectl Contrailstatus | 39

4 **Manage**

Manage Single Cluster CN2 | 45

- Overview | **45**
- Run Preflight and Postflight Checks | **45**
- Upgrade CN2 | **47**
- Uninstall CN2 | **48**

Manage Multi-Cluster CN2 | 49

- Attach a Workload Cluster | **50**
- Detach a Workload Cluster | **55**
- Uninstall CN2 | **56**

5 **Appendix**

Create a Rancher RKE2 Cluster | 59

- Configure a Server Node | **59**
- Configure an Agent Node | **63**

Configure Repository Credentials | 66

Prepare a Cluster Node for DPDK | 67

Juniper CN2 Technology Previews (Tech Previews) | 68

1

CHAPTER

Introduction

Cloud-Native Contrail Networking Overview | 2

Terminology | 4

CN2 Components | 6

Deployment Models | 11

System Requirements | 15

Cloud-Native Contrail Networking Overview

SUMMARY

Learn about Cloud-Native Contrail Networking (CN2).

IN THIS SECTION

- Benefits of Cloud-Native Contrail Networking | 4

NOTE: This section is intended to provide a brief overview of the Juniper Networks Cloud-Native Contrail Networking solution and might contain a description of features not supported in the Kubernetes distribution that you're using. See the Cloud-Native Contrail Networking Release Notes for information on features in the current release for your distribution.

Unless otherwise indicated, all references to Kubernetes in this Overview section are made generically and are not intended to single out a particular distribution.

In release 23.3, Cloud-Native Contrail Networking is supported on the following:

- (Upstream) Kubernetes
- Red Hat Openshift
- Amazon EKS
- Rancher RKE2

Contrail Networking is an SDN solution that automates the creation and management of virtualized networks to connect, isolate, and secure cloud workloads and services seamlessly across private and public clouds.

Cloud-Native Contrail Networking (CN2) brings this rich SDN feature set natively to Kubernetes as a networking platform and container network interface (CNI) plug-in.

Redesigned for cloud-native architectures, CN2 takes advantage of the benefits that Kubernetes offers, from simplified DevOps to turnkey scalability, all built on a highly available platform. These benefits include leveraging standard Kubernetes tools and practices to manage Contrail throughout its life cycle:

- Manage CN2 using standard Kubernetes and third-party tools.
- Scale CN2 by adding or removing nodes.
- Configure CN2 by using custom resource definitions (CRDs).

- Upgrade CN2 software by applying updated manifests.
- Uninstall CN2 by deleting Contrail namespaces and resources (where supported).

More than a CNI plug-in, CN2 is a networking platform that provides dynamic end-to-end virtual networking and security for cloud-native containerized and virtual machine (VM) workloads, across multi-cluster compute and storage environments, all from a central point of control. It supports hard multi-tenancy for single or multi-cluster environments shared across many tenants, teams, applications, or engineering phases, scaling to thousands of nodes.

The CN2 implementation consists of a set of Contrail controllers that reside on either Kubernetes control plane nodes or worker nodes depending on distribution. The Contrail controllers manage a distributed set of data planes implemented by a CNI plug-in and vRouter on every node. Integrating a full-fledged vRouter alongside the workloads provides CN2 the flexibility to support a wide range of networking requirements, from small single clusters to multi-cluster deployments, including:

- Full overlay networking including load balancing, security and multi-tenancy, elastic and resilient VPNs, and gateway services in single-cluster and multi-cluster deployments
- Highly available and resilient network controller overseeing all aspects of the network configuration and control planes
- Analytics services using telemetry and industry standard monitoring and presentation tools such as Prometheus and Grafana
- Support for both CRI-O and containerd runtimes
- Support for container and VM workloads (using kubevirt)
- Support for DPDK data plane acceleration

The Contrail controller automatically detects workload provisioning events such as a new workload being instantiated, network provisioning events such as a new virtual network being created, routing updates from internal and external sources, and unexpected network events such as link and node failures. The Contrail controller reports and logs these events where appropriate and reconfigures the vRouter data plane as necessary.

Although any single node can contain only one Contrail controller, a typical deployment contains multiple controllers running on multiple nodes. When there are multiple Contrail controllers, the controllers keep in synchronization by using iBGP to exchange routes. If a Contrail controller goes down, the Contrail controllers on the other nodes retain all database information and continue to provide the network control plane uninterrupted.

On the worker nodes where workloads reside, each vRouter establishes communications with two Contrail controllers, such that the vRouter can continue to receive instruction if any one controller goes down.

By natively supporting Kubernetes, the CN2 solution leverages the simplicity, flexibility, scalability, and availability inherent to the Kubernetes architecture, while supporting a rich SDN feature set that can meet the requirements of enterprises and service providers alike. Enterprises and service providers can now manage Contrail using simplified and familiar DevOps tools and processes without needing to learn a new life cycle management (LCM) paradigm.

Benefits of Cloud-Native Contrail Networking

- Support a rich networking feature set for your overlay networks.
- Deploy a highly scalable and highly available SDN solution on both upstream and commercial Kubernetes distributions.
- Manage CN2 using familiar, industry-standard tools and practices.
- Optionally, use the CN2 Web UI to configure and monitor your network.
- Leverage the skill set of your existing DevOps engineers to quickly get CN2 up and running.
- Combine with Juniper Networks fabric devices and fabric management solutions or use your own fabric or third-party cloud networks.

Terminology

Table 1: Terminology

Term	Meaning
Kubernetes control plane	The Kubernetes control plane is the collection of pods that manage containerized workloads on the worker nodes in a cluster.
Kubernetes control plane node	This is the virtual or physical machine that hosts the Kubernetes control plane, formerly known as a master node.
Server node	In Rancher terminology, a server node is a Kubernetes control plane node.

Table 1: Terminology *(Continued)*

Term	Meaning
Kubernetes node or worker node	Also called a worker node, a Kubernetes node is a virtual or physical machine that hosts containerized workloads in a cluster. To reduce ambiguity, we refer to this strictly as a worker node in this document.
Agent node	In Rancher terminology, an agent node is a Kubernetes worker node.
Contrail compute node	This is equivalent to a worker node. It is the node where the Contrail vRouter is providing the data plane function.
Network control plane	The network control plane provides the core SDN capability. It uses BGP to interact with peers such as other controllers and gateway routers, and XMPP to interact with the data plane components. CN2 supports a centralized network control plane architecture where the routing daemon runs centrally within the Contrail controller and learns and distributes routes from and to the data plane components. This centralized architecture facilitates virtual network abstraction, orchestration, and automation.
Network configuration plane	The network configuration plane interacts with Kubernetes control plane components to manage all CN2 resources. You configure CN2 resources using custom resource definitions (CRDs).
Network data plane	The network data plane resides on all nodes and interacts with containerized workloads to send and receive network traffic. Its main component is the Contrail vRouter.
Contrail controller	This is the part of CN2 that provides the network configuration and network control plane functionality. This name is purely conceptual – there is no corresponding Contrail controller object or entity in the UI.
Contrail controller node	This is the control plane node or worker node where the Contrail controller resides. In some Kubernetes distributions, the Contrail controller resides on control plane nodes. In other distributions, the Contrail controller resides on worker nodes.
Central cluster	In a multi-cluster deployment, this is the central Kubernetes cluster that houses the Contrail controller.

Table 1: Terminology *(Continued)*

Term	Meaning
Workload cluster	In a multi-cluster deployment, this is the distributed cluster that contains the workloads.

CN2 Components

The CN2 architecture consists of pods that perform the network configuration plane and network control plane functions, and pods that perform the network data plane functions.

- The network configuration plane refers to the functionality that enables CN2 to manage its resources and interact with the rest of the Kubernetes control plane.
- The network control plane represents CN2's full-featured SDN capability. It uses BGP to communicate with other controllers and XMPP to communicate with the distributed data plane components on the worker nodes.
- The network data plane refers to the packet transmit and receive function on every node, especially on worker nodes where the workloads reside.

The pods that perform the configuration and control plane functions reside on Kubernetes control plane nodes. The pods that perform the data plane functions reside on both Kubernetes control plane nodes and Kubernetes worker nodes.

Table 2 on page 7 describes the main CN2 components. Depending on configuration, there might be other components as well (not shown) that perform ancillary functions such as certificate management and status monitoring.

Table 2: CN2 Components

Pod Name		Where	Description
Configuration Plane[1]	contrail-k8s-apiserver	Control Plane Node	This pod is an aggregated API server that is the entry point for managing all Contrail resources. It is registered with the regular kube-apiserver as an APIService. The regular kube-apiserver forwards all network-related requests to the contrail-k8s-apiserver for handling. There is one contrail-k8s-apiserver pod per Kubernetes control plane node.
	contrail-k8s-controller	Control Plane Node	This pod performs the Kubernetes control loop function to reconcile networking resources. It constantly monitors networking resources to make sure the actual state of a resource matches its intended state. There is one contrail-k8s-controller pod per Kubernetes control plane node.
	contrail-k8s-kubemanager	Control Plane Node	This pod is the interface between Kubernetes resources and Contrail resources. It watches the kube-apiserver for changes to regular Kubernetes resources such as service and namespace and acts on any changes that affect the networking resources. In a single-cluster deployment, there is one contrail-k8s-kubemanager pod per Kubernetes control plane node. In a multi-cluster deployment, there is additionally one contrail-k8s-kubemanager pod for every distributed workload cluster.

Table 2: CN2 Components *(Continued)*

Pod Name		Where	Description
Control Plane[1]	contrail-control	Control Plane Node	This pod passes configuration to the worker nodes and performs route learning and distribution. It watches the kube-apiserver for anything affecting the network control plane and then communicates with its BGP peers and/or vRouter agents (over XMPP) as appropriate. There is one contrail-control pod per Kubernetes control plane node.
Data Plane	contrail-vrouter-nodes	Worker Node	This pod contains the vRouter agent and the vRouter itself. The vRouter agent acts on behalf of the local vRouter when interacting with the Contrail controller. There is one agent per node. The agent establishes XMPP sessions with two Contrail controllers to perform the following functions: • translates configuration from the control plane into objects that the vRouter understands • interfaces with the control plane for the management of routes • collects and exports statistics from the data plane The vRouter provides the packet send and receive function for the co-located pods and workloads. It provides the CNI plug-in functionality.
	contrail-vrouter-masters	Control Plane Node	This pod provides the same functionality as the contrail-vrouter-nodes pod, but resides on the control plane nodes.

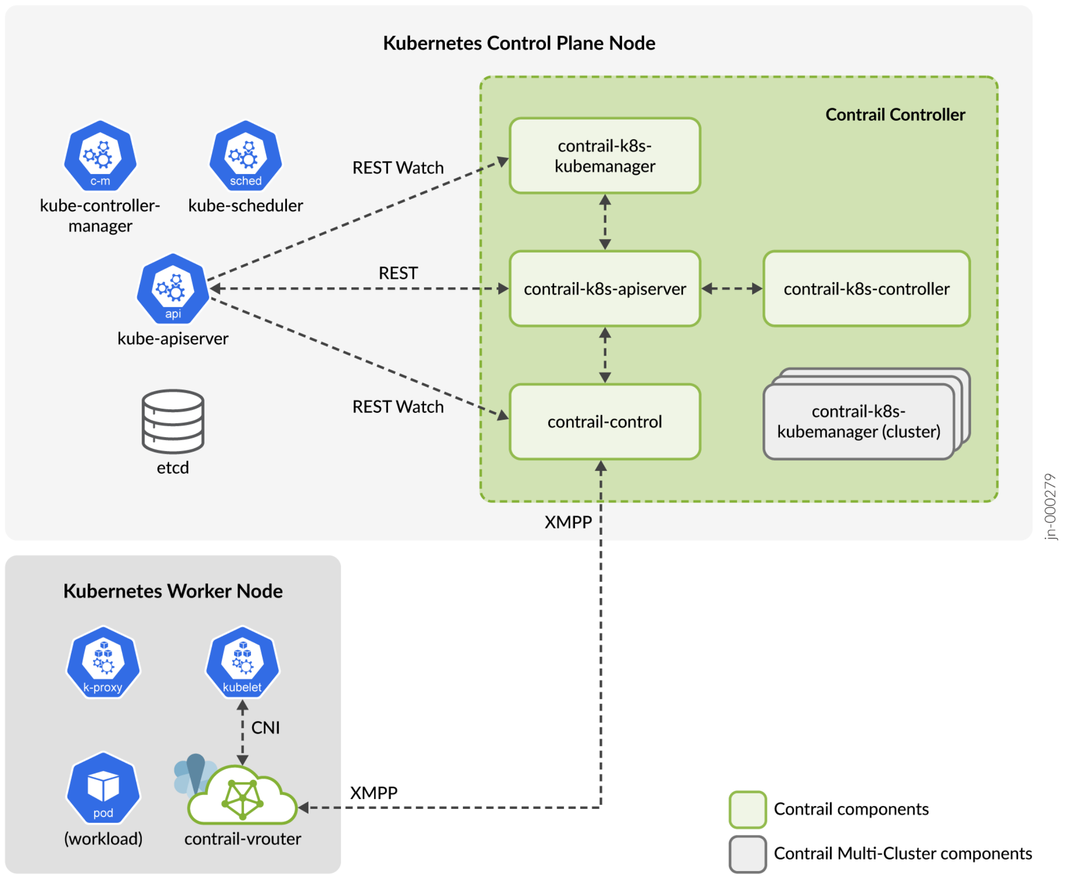
Table 2: CN2 Components *(Continued)*

Pod Name	Where	Description
[1]The components that make up the network configuration plane and the network control plane are collectively called the Contrail controller.		

Figure 1 on page 9 shows these components in the context of a Kubernetes cluster.

For clarity and to reduce clutter, the figures do not show the data plane pods on the node with the Contrail controller.

Figure 1: CN2 Components



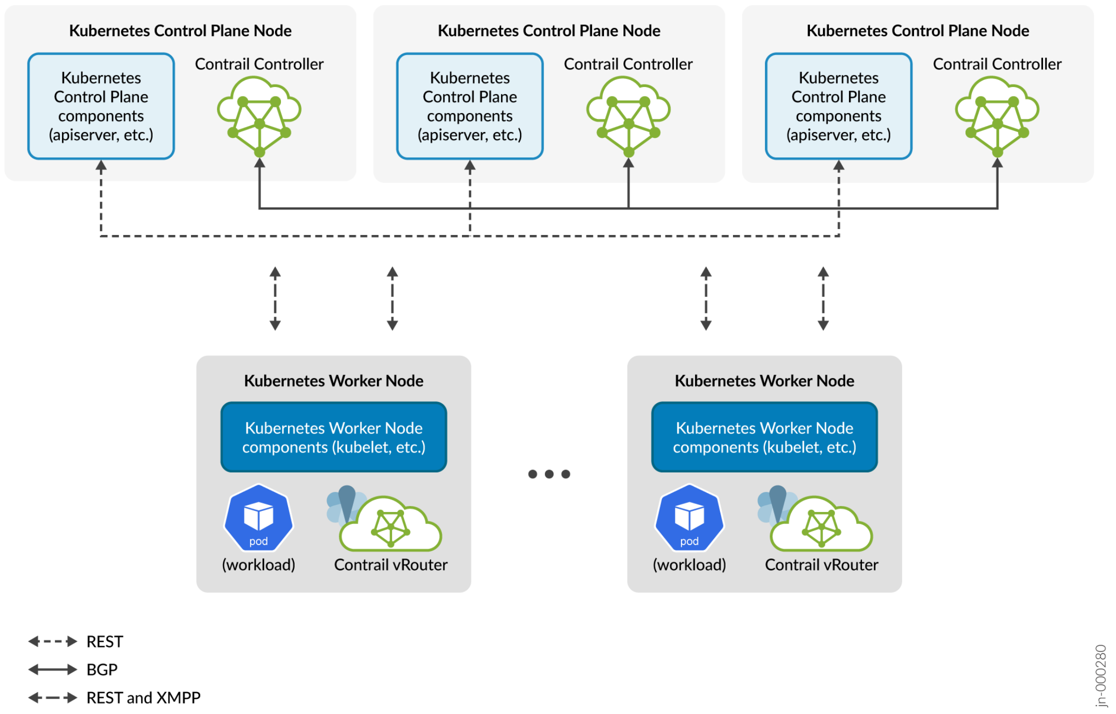
When running on upstream Kubernetes or Rancher RKE2, the Contrail controller stores all CN2 cluster data in the main Kubernetes etcd database by default. When running on OpenShift, the Contrail controller stores all CN2 cluster data in its own Contrail etcd database.

The kube-apiserver is the entry point for Kubernetes REST API calls for the cluster. It directs all networking requests to the contrail-k8s-apiserver, which is the entry point for Contrail API calls. The contrail-k8s-apiserver translates incoming networking requests into REST API calls to the respective CN2 objects. In some cases, these calls may result in the Contrail controller sending XMPP messages to the vRouter agent on one or more worker nodes or sending BGP messages (not shown) to other control plane nodes or external routers. These XMPP and BGP messages are sent outside of regular Kubernetes node-to-node communications.

The contrail-k8s-kubemanager (cluster) components are only present in multi-cluster deployments. For more information on the different types of deployment, see Deployment Models.

Figure 2 on page 10 shows a cluster with multiple Contrail controllers. These controllers reside on control plane nodes. The Kubernetes components communicate with each other using REST. The Contrail controllers exchange routes with each other using iBGP, outside of the regular Kubernetes REST interface. For redundancy, the vRouter agents on worker nodes always establish XMPP communications with two Contrail controllers.

Figure 2: Multiple Contrail Controllers


Kubernetes Control Plane Node
Kubernetes Control Plane components (apiserver, etc.)
Contrail Controller
Kubernetes Control Plane Node
Kubernetes Control Plane components (apiserver, etc.)
Contrail Controller
Kubernetes Control Plane Node
Kubernetes Control Plane components (apiserver, etc.)
Contrail Controller
Kubernetes Worker Node
Kubernetes Worker Node components (kubelet, etc.)
pod
(workload)
Contrail vRouter
Kubernetes Worker Node
Kubernetes Worker Node components (kubelet, etc.)
pod
(workload)
Contrail vRouter
REST
BGP
REST and XMPP
jn-000280

Deployment Models

SUMMARY

Learn about single cluster and multi-cluster CN2.

IN THIS SECTION

- Single Cluster Deployment | 11
- Multi-Cluster Deployment | 12

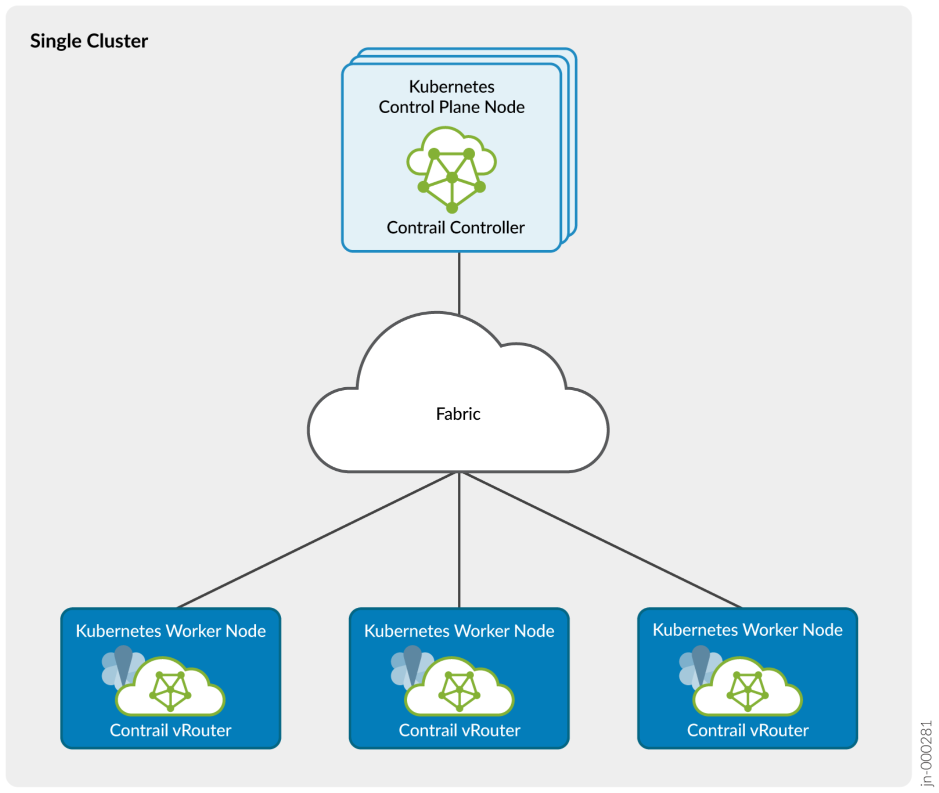
Cloud-Native Contrail Networking (CN2) is available both as an integrated networking platform in a single Kubernetes cluster and as a centralized networking platform to multiple distributed Kubernetes clusters. In both cases, Contrail works as an integrated component of your infrastructure by watching where workloads are instantiated and connecting those workloads to the appropriate overlay networks.

Single Cluster Deployment

Cloud-Native Contrail Networking (CN2) is available as an integrated networking platform in a single Kubernetes cluster, watching where workloads are instantiated and connecting those workloads to the appropriate overlay networks.

In a single-cluster deployment (Figure 3 on page 12), the Contrail controller sits in the Kubernetes control plane and provides the network configuration and network control planes for the host cluster. The Contrail data plane components sit in all nodes and provide the packet send and receive function for the workloads.

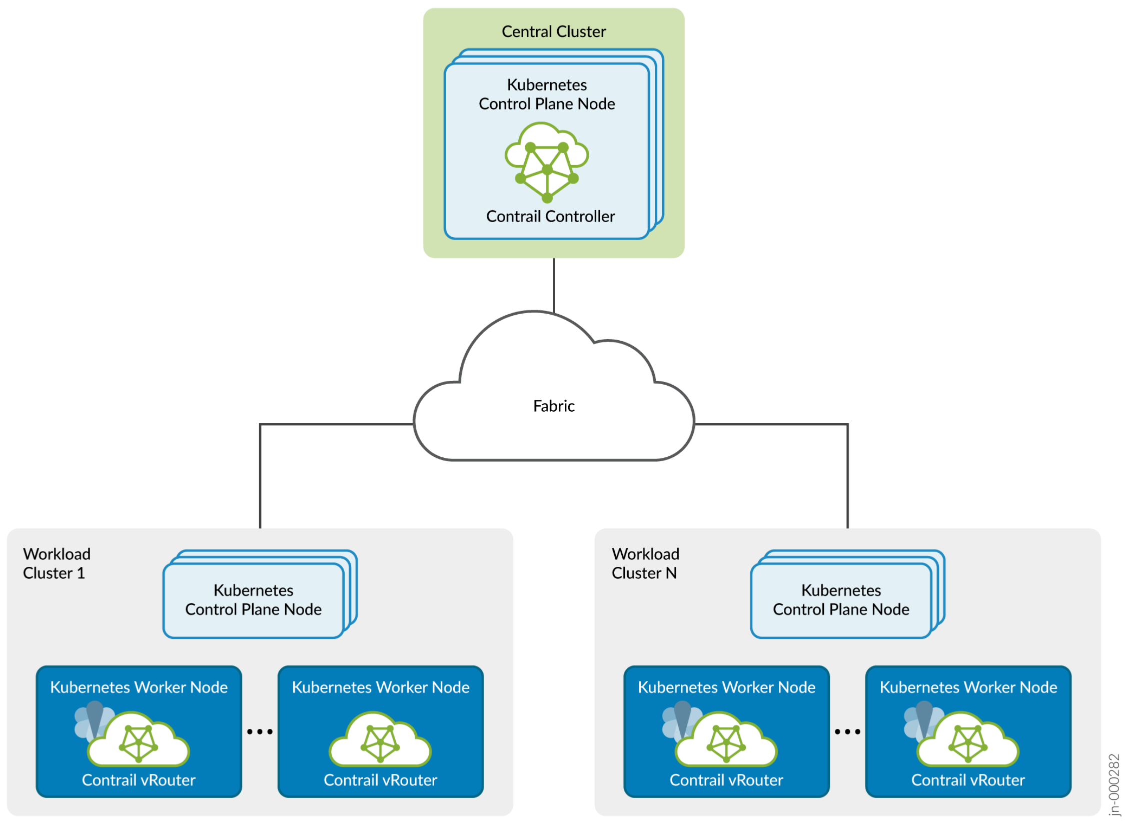
Figure 3: Single Cluster Deployment



Multi-Cluster Deployment

In a multi-cluster deployment (Figure 4 on page 13), the Contrail controller resides in its own Kubernetes cluster and provides networking to other clusters. The Kubernetes cluster that the Contrail controller resides in is called the central cluster. The Kubernetes clusters that house the workloads are called the distributed workload clusters.

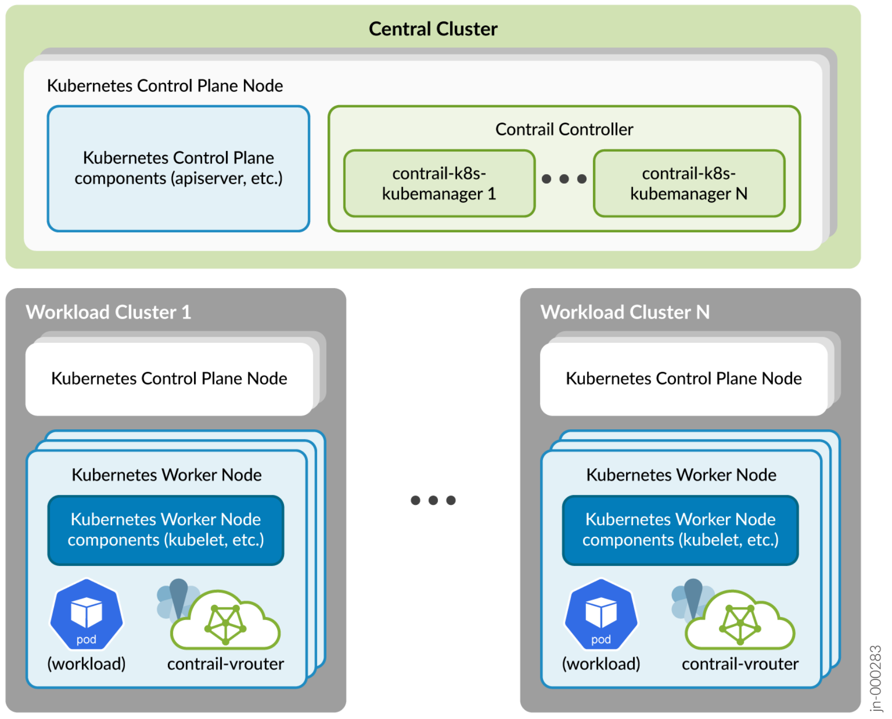
Figure 4: Multi-Cluster Deployment



Centralizing the network function in this way makes it not only easier to configure and manage, but also easier to apply consistent network policy and security.

Figure 5 on page 14 provides more detail on this setup. The Contrail controller sits in the Kubernetes control plane of the central cluster and contains a kubemanager for each workload cluster that it serves. There are typically no worker nodes in the central cluster. Instead, the workloads reside in the worker nodes in the distributed workload clusters. The Contrail CNI plugin and vRouter sit in the worker nodes of the workload clusters. The Kubernetes control plane in the workload clusters do not contain any Contrail controller components.

Figure 5: Multi-Cluster Components


Central Cluster
Kubernetes Control Plane Node
Kubernetes Control Plane components (apiserver, etc.)
Contrail Controller
contrail-k8s-kubemanager 1
contrail-k8s-kubemanager N
Workload Cluster 1
Kubernetes Control Plane Node
Kubernetes Worker Node
Kubernetes Worker Node components (kubelet, etc.)
pod
(workload)
contrail-vrouter
Workload Cluster N
Kubernetes Control Plane Node
Kubernetes Worker Node
Kubernetes Worker Node components (kubelet, etc.)
pod
(workload)
contrail-vrouter
jn-000283

The multi-cluster Contrail controller differs from the single-cluster Contrail controller in two main ways:

- The multi-cluster Contrail controller has a contrail-k8s-kubemanager pod instantiated for each distributed workload cluster. As part of the procedure to connect a distributed workload cluster to the central cluster, you explicitly create and assign a contrail-k8s-kubemanager deployment that watches for changes to resources that affect its assigned workload cluster.

- The multi-cluster Contrail controller uses multi-cluster watch technology to detect changes in the distributed workload clusters.

The function of the multi-cluster contrail-k8s-kubemanager pod is identical to its single-cluster counterpart. It watches for changes to regular Kubernetes resources that affect its assigned cluster and acts on the changes accordingly.

All other Contrail components in a multi-cluster deployment behave in the same way as in a single-cluster deployment. The network control plane, for example, communicates with data plane components using XMPP, outside of regular Kubernetes REST channels. Because of this, the network control plane is indifferent to whether the data plane components that it communicates with reside in the same cluster or in different clusters. The only requirement is that the data plane components are reachable.

System Requirements

Table 3: System Requirements for Rancher RKE2 Installation with CN2

Machine	CPU	RAM	Storage	Notes
Control Plane (Server) Nodes [1]	8	32 GB	400 GB	Processor must support the AVX2 instruction set if running DPDK.
Worker (Agent) Nodes[2]	4	16 GB	100 GB	Processor must support the AVX2 instruction set if running DPDK.

[1] Includes nodes in single clusters, central clusters, and distributed workload clusters.

[2] Based on workload requirements.

2

CHAPTER

Install

Overview | 17

Before You Install | 18

Install Single Cluster CN2 on Rancher RKE2 | 19

Install Multi-Cluster CN2 on Rancher RKE2 | 28

Install Contrail Tools | 29

Manifests | 31

Overview

IN THIS SECTION

- Benefits of Rancher RKE2 with CN2 | 17

Rancher is a complete container management platform that lets you deploy and run clusters on any provider. Rancher lets you deploy and manage custom Kubernetes clusters with a wide range of features that simplify the deployment, orchestration, and scaling of containerized applications. With Rancher, users can easily manage multiple Kubernetes clusters across different cloud providers. Rancher provides centralized authentication, role-based access control (RBAC), and monitoring capabilities. This makes it easier to collaborate and manage containerized applications.

Rancher Kubernetes Engine 2 (RKE2), is a lightweight Kubernetes distribution that is specifically designed for security and compliance. RKE2 builds upon RKE and offers additional enhancements for improved performance and security. RKE2 is Cloud Native Computing Foundation (CNCF) certified and offers Federal Information Processing Standard (FIPS) 14-2 compliance. RKE2 leverages containerd as the container runtime, reducing resource overhead and improving performance. RKE2 also supports high availability (HA) deployments to ensure reliability and application redundancy.

Rancher and RKE2 provide a robust range of features for managing and deploying Kubernetes clusters. Rancher's interface makes it easy to manage multiple RKE2 clusters, whether they are located in the cloud, on-premises, or at the edge. RKE2's lightweight design and high availability capabilities makes it an ideal choice for deploying Kubernetes in a range of environments.

Environments running a Rancher RKE2 cluster with CN2 as the CNI benefit from a feature-rich CNI platform with advanced capabilities. CN2 offers features like network segmentation and isolation, seamless pod-to-pod connectivity within the cluster, and multi-tenancy. By deploying CN2 as the CNI of your Rancher RKE2 cluster, you can leverage the benefits of both platforms.

Due to the flexibility of both CN2 and RKE2, installing an RKE2 cluster with CN2 as the CNI is relatively straightforward. The Rancher installation process is streamlined and requires minimal configuration. After your RKE2 cluster is up, install CN2 by applying the CN2 manifests.

Benefits of Rancher RKE2 with CN2

- Ease-of-use and flexibility with a streamlined installation process for speedy deployments

- Enhanced networking capabilities with CN2-provided features like Virtual Routing and Forwarding (VRF), and Rancher RKE2-provided features like centralized RBAC
- Industry-leading SDN solution that emphasizes centralized control and scalability

Before You Install

1. Set up an account with Juniper Networks so you can download CN2 manifests from the Juniper Networks download site (https://support.juniper.net/support/downloads/?p=contrail-networking) and access the container repository at https://enterprise-hub.juniper.net.
2. Set up the fabric network and connect your nodes to the fabric.

   The example networks used in this document are shown in the respective installation sections.
3. Configure the server and agent (cluster) nodes.

   a. Install a fresh OS on all servers/VMs that you'll use as cluster nodes. Ensure the OS and kernel versions on the cluster nodes are on the list of supported OSes and kernels (see the *CN2 Tested Integrations* matrix at https://www.juniper.net/documentation/us/en/software/cn-cloud-native/cn2-tested-integrations/cn-cloud-native-tested-integrations/concept/cn-cloud-native-tested-integrations.html).

   b. Configure the OS on each node minimally for the following:

   - static IP address and mask as per the example cluster you want to install (for example, 172.16.0.11/24 through 172.16.0.13/24 in our single cluster example) and gateway
   - access to one or more DNS servers

     NOTE: If you're running **systemd-resolved** on Ubuntu, ensure that **/etc/resolv.conf** is linked to **/run/systemd/resolve/resolv.conf**, and not to **/run/systemd/resolve/stub-resolv.conf**.

     ```
     ln -svf /run/systemd/resolve/resolv.conf /etc/resolv.conf


     ls -l /etc/resolv.conf
     lrwxrwxrwx 1 root root 32 May 23 19:15 /etc/resolv.conf -> /run/systemd/resolve/
     resolv.conf
     ```

     https://kubernetes.io/docs/tasks/administer-cluster/dns-debugging-resolution/#known-issues

- SSH connectivity including root SSH access
- NTP (must be chrony)

The cluster nodes in our examples are running Ubuntu.

c. If you're planning on running with a DPDK data plane, prepare each cluster node that is running DPDK.

   For an example on how to do this, see "Prepare a Cluster Node for DPDK" on page 67.

4. Download the Contrail Networking manifests ("Manifests" on page 31) and extract the tgz onto the host where you plan on running the installation. This host must be able to reach the cluster nodes.

   In the examples in this document, we run the installation on an RKE2 server node.

5. Configure your repository login credentials in the downloaded manifests.

   Add your repository login credentials to the **k8s** and **contrail-tools** manifests. See "Configure Repository Credentials" on page 66 for one way to do this.

6. Install Contrail tools.

   See "Install Contrail Tools" on page 29.

7. Install contrailstatus on the machine where you plan on running kubectl. Contrailstatus is a kubectl plug-in you can use to query Contrail microservices and Contrail-specific resources.

   In the examples in this document, we run kubectl on an RKE2 server node.

   The contrailstatus executable is packaged within the downloaded tools package. Extract and copy the **kubectl-contrailstatus** executable to **/usr/local/bin**.

If you're installing a multi-cluster, then repeat steps 4 to 7 for each cluster.

Install Single Cluster CN2 on Rancher RKE2

SUMMARY

See examples on how to install single cluster CN2 on Rancher RKE2.

IN THIS SECTION

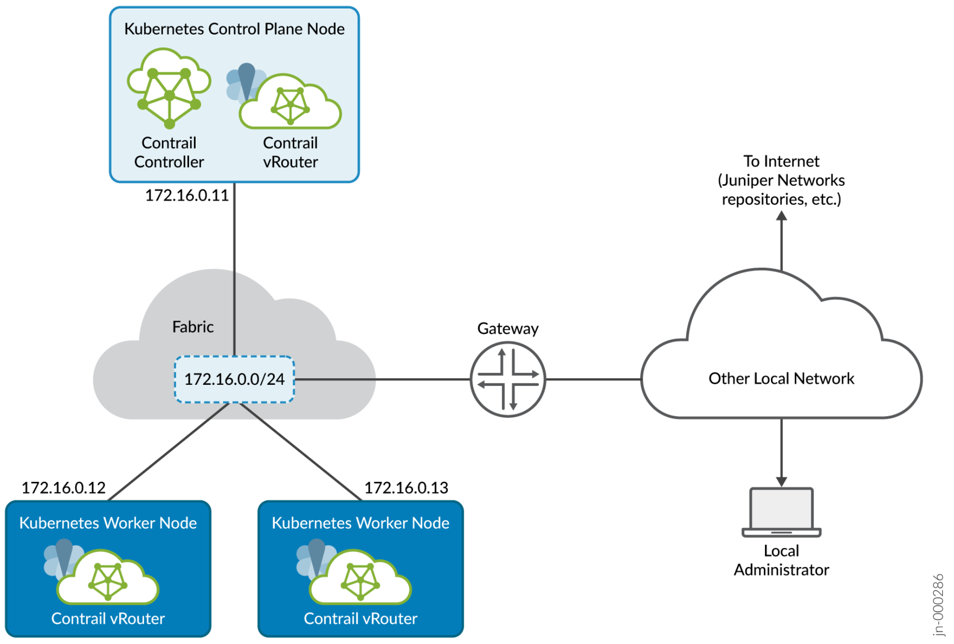
- Install Single Cluster CN2 on Rancher RKE2 Running Kernel Mode Data Plane | 21
- Install Single Cluster CN2 on Rancher RKE2 Running DPDK Data Plane | 24

In a single cluster shared network deployment, CN2 is the networking platform and CNI plug-in for that cluster. The Contrail controller runs in the Kubernetes control plane, and the Contrail data plane components run on all nodes in the cluster.

Figure 6 on page 20 shows the cluster that you'll create if you follow this single cluster example. The cluster consists of a single server (control plane) node and two agent (worker) nodes.

All nodes shown can be VMs or bare metal servers.

Figure 6: Single Cluster CN2 on Rancher RKE2



All communication between nodes in the cluster and between nodes and external sites takes place over the single 172.16.0.0/24 fabric virtual network. The fabric network provides the underlay over which the cluster runs.

The local administrator is shown attached to a separate network reachable through a gateway. This is typical of many installations where the local administrator manages the fabric and cluster from the corporate LAN. In the procedures that follow, we refer to the local administrator station as your local computer.

NOTE: Connecting all cluster nodes together is the data center fabric, which is shown in the example as a single subnet. In real installations, the data center fabric is a network of spine and leaf switches that provide the physical connectivity for the cluster.

In an Apstra-managed data center, this connectivity would be specified through the overlay virtual networks that you create across the underlying fabric switches.

The procedures in this section show basic examples of how you can use the provided manifests to create the specified CN2 deployment. You're not limited to the deployment described in this section nor are you limited to using the provided manifests. CN2 supports a wide range of deployments that are too numerous to cover in detail. Use the provided examples as a starting point to roll your own manifest tailored to your specific situation.

Install Single Cluster CN2 on Rancher RKE2 Running Kernel Mode Data Plane

Use this procedure to install CN2 in a single cluster deployment running a kernel mode data plane.

The manifest that you will use in this example procedure is **k8s/single-cluster/single_cluster_deployer_example.yaml**. The procedure assumes that you've placed this manifest into a **manifests** directory.

To install CN2 on a Rancher RKE2 cluster, you first create a Rancher RKE2 cluster with no CNI. Then you apply the CN2 deployer manifest, which installs CN2 onto the cluster.

1. Create a Rancher RKE2 cluster. You can follow the example procedure in "Create a Rancher RKE2 Cluster" on page 59 or you can use any other method. Create the cluster with no CNI plug-in.
2. Apply the Contrail deployer manifest.

   ```
   kubectl apply -f manifests/single_cluster_deployer_example.yaml
   ```

   It may take a few minutes for the nodes and pods to come up.
3. Use standard kubectl commands to check on the deployment.

a. Show the status of the nodes.

```
kubectl get nodes
```

```
NAME      STATUS   ROLES                       AGE   VERSION
rke2-a1   Ready    <none>                      17h   v1.25.10+rke2r1
rke2-a2   Ready    <none>                      17h   v1.25.10+rke2r1
rke2-s1   Ready    control-plane,etcd,master   17h   v1.25.10+rke2r1
```

You can see that the nodes are now up. If the nodes are not up, wait a few minutes and check again.

b. Show the status of the pods.

```
kubectl get pods -A -o wide
```

```
NAMESPACE         NAME                                                  READY   STATUS     RESTARTS
AGE   IP            NODE      NOMINATED NODE   READINESS GATES
cert-manager      cert-manager-6dc787b997-cfbvp                         1/1     Running    0
12h   172.16.0.12   rke2-a1   <none>           <none>
cert-manager      cert-manager-cainjector-6f96556ddf-spqpb              1/1     Running    0
12h   172.16.0.13   rke2-a2   <none>           <none>
cert-manager      cert-manager-webhook-9d965dff5-cllfm                  1/1     Running    0
12h   172.16.0.12   rke2-a1   <none>           <none>
contrail-deploy   contrail-k8s-deployer-f8cd78888-pmgpl                 1/1     Running    1 (15h ago)
15h   172.16.0.11   rke2-s1   <none>           <none>
contrail-system   contrail-k8s-apiserver-5d458f8d69-7s9nb               1/1     Running    0
12h   172.16.0.11   rke2-s1   <none>           <none>
contrail-system   contrail-k8s-controller-cb4bc88cc-xc2kx               1/1     Running    0
12h   172.16.0.11   rke2-s1   <none>           <none>
contrail          contrail-control-0                                    2/2     Running    0
12h   172.16.0.11   rke2-s1   <none>           <none>
contrail          contrail-k8s-contrailstatusmonitor-5497bb64b7-dztrz   1/1     Running    0
12h   172.16.0.11   rke2-s1   <none>           <none>
contrail          contrail-k8s-kubemanager-7d7f5b4c88-rgk2k             1/1     Running    0
12h   172.16.0.11   rke2-s1   <none>           <none>
contrail          contrail-vrouter-masters-gxlcs                        3/3     Running    0
12h   172.16.0.11   rke2-s1   <none>           <none>
contrail          contrail-vrouter-nodes-mbgf9                          3/3     Running    0
```

```
12h   172.16.0.13   rke2-a2   <none>           <none>
contrail          contrail-vrouter-nodes-mmnf7                         3/3     Running     0
12h   172.16.0.12   rke2-a1   <none>           <none>
kube-system       cloud-controller-manager-rke2-s1                     1/1     Running     0
17h   172.16.0.11   rke2-s1   <none>           <none>
kube-system       etcd-rke2-s1                                         1/1     Running     0
17h   172.16.0.11   rke2-s1   <none>           <none>
kube-system       helm-install-rke2-coredns-trcjf                      0/1     Completed   0
17h   172.16.0.11   rke2-s1   <none>           <none>
kube-system       helm-install-rke2-ingress-nginx-jghfq                0/1     Completed   11
17h   10.42.0.6     rke2-s1   <none>           <none>
kube-system       helm-install-rke2-metrics-server-nvcp6               0/1     Completed   0
17h   10.42.0.3     rke2-s1   <none>           <none>
kube-system       helm-install-rke2-snapshot-controller-9gbf7          0/1     Completed   3
17h   10.42.0.7     rke2-s1   <none>           <none>
kube-system       helm-install-rke2-snapshot-controller-crd-l5wnn      0/1     Completed   0
17h   10.42.0.4     rke2-s1   <none>           <none>
kube-system       helm-install-rke2-snapshot-validation-webhook-kqmnp  0/1     Completed   0
17h   10.42.0.5     rke2-s1   <none>           <none>
kube-system       kube-apiserver-rke2-s1                               1/1     Running     0
17h   172.16.0.11   rke2-s1   <none>           <none>
kube-system       kube-controller-manager-rke2-s1                      1/1     Running     0
17h   172.16.0.11   rke2-s1   <none>           <none>
kube-system       kube-proxy-rke2-a1                                   1/1     Running     0
17h   172.16.0.12   rke2-a1   <none>           <none>
kube-system       kube-proxy-rke2-a2                                   1/1     Running     0
17h   172.16.0.13   rke2-a2   <none>           <none>
kube-system       kube-proxy-rke2-s1                                   1/1     Running     0
17h   172.16.0.11   rke2-s1   <none>           <none>
kube-system       kube-scheduler-rke2-s1                               1/1     Running     0
17h   172.16.0.11   rke2-s1   <none>           <none>
kube-system       rke2-coredns-rke2-coredns-6b9548f79f-gqpzl           1/1     Running     0
12h   10.42.2.1     rke2-a2   <none>           <none>
kube-system       rke2-coredns-rke2-coredns-6b9548f79f-snzl4           1/1     Running     0
17h   10.42.0.8     rke2-s1   <none>           <none>
kube-system       rke2-coredns-rke2-coredns-autoscaler-57647bc7cf-dc2gx   1/1  Running     0
17h   10.42.0.2     rke2-s1   <none>           <none>
kube-system       rke2-ingress-nginx-controller-6sk9w                  1/1     Running     0
11h   10.42.0.9     rke2-s1   <none>           <none>
kube-system       rke2-ingress-nginx-controller-ng4hg                  1/1     Running     0
11h   10.42.2.3     rke2-a2   <none>           <none>
kube-system       rke2-ingress-nginx-controller-rrrts                  1/1     Running     0
11h   10.42.1.0     rke2-a1   <none>           <none>
```

```
kube-system      rke2-metrics-server-78b84fff48-jvnnd                1/1     Running     0
12h   10.42.2.0     rke2-a2   <none>           <none>
kube-system      rke2-snapshot-controller-849d69c748-v42dv           1/1     Running     0
12h   10.42.2.2     rke2-a2   <none>           <none>
kube-system      rke2-snapshot-validation-webhook-654f6677b-c4v5z    1/1     Running     0
12h   10.42.1.1     rke2-a1   <none>           <none>
```

All pods should now have a STATUS of Running. If not, wait a few minutes for the pods to come up.

c. If some pods remain down, debug the deployment as you normally do. Use the `kubectl describe` command to see why a pod is not coming up. A common error is a network or firewall issue preventing the node from reaching the Juniper Networks repository.

   Here is an example of a DNS problem.

   Log in to each node having a problem and check name resolution for enterprise-hub.juniper.net. For example:

```
ping enterprise-hub.juniper.net
ping: enterprise-hub.juniper.net: Temporary failure in name resolution
```

NOTE: Although enterprise-hub.juniper.net is not configured to respond to pings, we can use the ping command to check domain name resolution.

   In this example, the domain name is not resolving. Check the domain name server configuration to make sure it's correct.

   For example, in a Ubuntu system running **systemd resolved**, check that **/etc/resolv.conf** is linked to **/run/systemd/resolve/resolv.conf** as described in step 3 in "Before You Install" on page 18 and check that your DNS server is listed correctly in that file.

d. If you run into a problem you can't solve or if you made a mistake during the installation, simply uninstall CN2 and start over. To uninstall CN2, see "Uninstall CN2" on page 48.

4. (Optional) Run postflight checks. See "Run Preflight and Postflight Checks" on page 45.

Install Single Cluster CN2 on Rancher RKE2 Running DPDK Data Plane

Use this procedure to install CN2 in a single cluster deployment running a DPDK data plane.

The manifest that you will use in this example procedure is **k8s/single-cluster/single_cluster_deployer_example.yaml**. The procedure assumes that you've placed this manifest into a **manifests** directory.

To install CN2 on a Rancher RKE2 cluster running DPDK, you first create a Rancher RKE2 cluster with no CNI. Then you label the DPDK nodes and apply the CN2 deployer manifest, which installs CN2 onto the cluster.

1. Create a Rancher RKE2 cluster. You can follow the example procedure in "Create a Rancher RKE2 Cluster" on page 59 or you can use any other method. Create the cluster with the following characteristics:
   - Cluster has no CNI plug-in.
   - Enable multus version 0.3.1.
2. Specify the DPDK nodes.

   For each node running DPDK, label it as follows:

   ```
   kubectl label node <node-name> agent-mode=dpdk
   ```

   By labeling the nodes in this way, CN2 will use the DPDK configuration specified in the manifest.
3. Apply the Contrail deployer manifest.

   ```
   kubectl apply -f manifests/single_cluster_deployer_example.yaml
   ```

   It may take a few minutes for the nodes and pods to come up.
4. Use standard kubectl commands to check on the deployment.

   a. Show the status of the nodes.

   ```
   NAME      STATUS   ROLES                       AGE   VERSION
   rke2-a1   Ready    <none>                      17h   v1.25.10+rke2r1
   rke2-a2   Ready    <none>                      17h   v1.25.10+rke2r1
   rke2-s1   Ready    control-plane,etcd,master   17h   v1.25.10+rke2r1
   ```

   You can see that the nodes are now up. If the nodes are not up, wait a few minutes and check again.

   b. Show the status of the pods.

   ```
   NAMESPACE         NAME                                                      READY   STATUS      RESTARTS
   AGE    IP             NODE      NOMINATED NODE    READINESS GATES
   ```

```
cert-manager      cert-manager-6dc787b997-cfbvp                          1/1     Running     0
12h   172.16.0.12   rke2-a1   <none>           <none>
cert-manager      cert-manager-cainjector-6f96556ddf-spqpb               1/1     Running     0
12h   172.16.0.13   rke2-a2   <none>           <none>
cert-manager      cert-manager-webhook-9d965dff5-cllfm                   1/1     Running     0
12h   172.16.0.12   rke2-a1   <none>           <none>
contrail-deploy   contrail-k8s-deployer-f8cd78888-pmgpl                  1/1     Running     1 (15h ago)
15h   172.16.0.11   rke2-s1   <none>           <none>
contrail-system   contrail-k8s-apiserver-5d458f8d69-7s9nb                1/1     Running     0
12h   172.16.0.11   rke2-s1   <none>           <none>
contrail-system   contrail-k8s-controller-cb4bc88cc-xc2kx                1/1     Running     0
12h   172.16.0.11   rke2-s1   <none>           <none>
contrail          contrail-control-0                                     2/2     Running     0
12h   172.16.0.11   rke2-s1   <none>           <none>
contrail          contrail-k8s-contrailstatusmonitor-5497bb64b7-dztrz    1/1     Running     0
12h   172.16.0.11   rke2-s1   <none>           <none>
contrail          contrail-k8s-kubemanager-7d7f5b4c88-rgk2k              1/1     Running     0
12h   172.16.0.11   rke2-s1   <none>           <none>
contrail          contrail-vrouter-masters-gxlcs                         3/3     Running     0
12h   172.16.0.11   rke2-s1   <none>           <none>
contrail          contrail-vrouter-nodes-mbgf9                           3/3     Running     0
12h   172.16.0.13   rke2-a2   <none>           <none>
contrail          contrail-vrouter-nodes-mmnf7                           3/3     Running     0
12h   172.16.0.12   rke2-a1   <none>           <none>
kube-system       cloud-controller-manager-rke2-s1                       1/1     Running     0
17h   172.16.0.11   rke2-s1   <none>           <none>
kube-system       etcd-rke2-s1                                           1/1     Running     0
17h   172.16.0.11   rke2-s1   <none>           <none>
kube-system       helm-install-rke2-coredns-trcjf                        0/1     Completed   0
17h   172.16.0.11   rke2-s1   <none>           <none>
kube-system       helm-install-rke2-ingress-nginx-jghfq                  0/1     Completed   11
17h   10.42.0.6     rke2-s1   <none>           <none>
kube-system       helm-install-rke2-metrics-server-nvcp6                 0/1     Completed   0
17h   10.42.0.3     rke2-s1   <none>           <none>
kube-system       helm-install-rke2-snapshot-controller-9gbf7            0/1     Completed   3
17h   10.42.0.7     rke2-s1   <none>           <none>
kube-system       helm-install-rke2-snapshot-controller-crd-l5wnn        0/1     Completed   0
17h   10.42.0.4     rke2-s1   <none>           <none>
kube-system       helm-install-rke2-snapshot-validation-webhook-kqmnp    0/1     Completed   0
17h   10.42.0.5     rke2-s1   <none>           <none>
kube-system       kube-apiserver-rke2-s1                                 1/1     Running     0
17h   172.16.0.11   rke2-s1   <none>           <none>
kube-system       kube-controller-manager-rke2-s1                        1/1     Running     0
```

```
17h   172.16.0.11   rke2-s1   <none>           <none>
kube-system       kube-proxy-rke2-a1                                     1/1     Running     0
17h   172.16.0.12   rke2-a1   <none>           <none>
kube-system       kube-proxy-rke2-a2                                     1/1     Running     0
17h   172.16.0.13   rke2-a2   <none>           <none>
kube-system       kube-proxy-rke2-s1                                     1/1     Running     0
17h   172.16.0.11   rke2-s1   <none>           <none>
kube-system       kube-scheduler-rke2-s1                                 1/1     Running     0
17h   172.16.0.11   rke2-s1   <none>           <none>
kube-system       rke2-coredns-rke2-coredns-6b9548f79f-gqpzl             1/1     Running     0
12h   10.42.2.1     rke2-a2   <none>           <none>
kube-system       rke2-coredns-rke2-coredns-6b9548f79f-snzl4             1/1     Running     0
17h   10.42.0.8     rke2-s1   <none>           <none>
kube-system       rke2-coredns-rke2-coredns-autoscaler-57647bc7cf-dc2gx  1/1     Running     0
17h   10.42.0.2     rke2-s1   <none>           <none>
kube-system       rke2-ingress-nginx-controller-6sk9w                    1/1     Running     0
11h   10.42.0.9     rke2-s1   <none>           <none>
kube-system       rke2-ingress-nginx-controller-ng4hg                    1/1     Running     0
11h   10.42.2.3     rke2-a2   <none>           <none>
kube-system       rke2-ingress-nginx-controller-rrrts                    1/1     Running     0
11h   10.42.1.0     rke2-a1   <none>           <none>
kube-system       rke2-metrics-server-78b84fff48-jvnnd                   1/1     Running     0
12h   10.42.2.0     rke2-a2   <none>           <none>
kube-system       rke2-snapshot-controller-849d69c748-v42dv              1/1     Running     0
12h   10.42.2.2     rke2-a2   <none>           <none>
kube-system       rke2-snapshot-validation-webhook-654f6677b-c4v5z       1/1     Running     0
12h   10.42.1.1     rke2-a1   <none>           <none>
```

All pods should now have a STATUS of Running. If not, wait a few minutes for the pods to come up.

c. If some pods remain down, debug the deployment as you normally do. Use the `kubectl describe` command to see why a pod is not coming up. A common error is a network or firewall issue preventing the node from reaching the Juniper Networks repository.

   Here is an example of a DNS problem.

   Log in to each node having a problem and check name resolution for enterprise-hub.juniper.net. For example:

```
ping enterprise-hub.juniper.net
ping: enterprise-hub.juniper.net: Temporary failure in name resolution
```

NOTE: Although enterprise-hub.juniper.net is not configured to respond to pings, we can use the ping command to check domain name resolution.

In this example, the domain name is not resolving. Check the domain name server configuration to make sure it's correct.

For example, in a Ubuntu system running **systemd resolved**, check that **/etc/resolv.conf** is linked to **/run/systemd/resolve/resolv.conf** as described in step 3 in "Before You Install" on page 18 and check that your DNS server is listed correctly in that file.

d. If you run into a problem you can't solve or if you made a mistake during the installation, simply uninstall CN2 and start over. To uninstall CN2, see "Uninstall CN2" on page 48.

5. (Optional) Run postflight checks. See "Run Preflight and Postflight Checks" on page 45.

Install Multi-Cluster CN2 on Rancher RKE2

Use this procedure to install CN2 in a multi-cluster deployment.

In a multi-cluster deployment, CN2 is the central networking platform and CNI plug-in for multiple distributed workload clusters. The Contrail controller runs in the Kubernetes control plane in the central cluster, and the Contrail data plane components run on the worker nodes in the distributed workload clusters.

To install CN2 in a multi-cluster deployment, you first create the central cluster and then you attach the distributed workload clusters to the central cluster one by one. As with the single-cluster deployment, you'll start with a fresh Rancher RKE2 cluster with no CNI plug-in installed and then you'll install CN2 on it.

The manifest that you will use to create the central cluster in this example procedure is **k8s/multi-cluster/central_cluster_deployer_example.yaml**. The procedure assumes that you've placed this manifest into a **manifests** directory.

The procedures in this section show basic examples of how you can use the provided manifests to create the specified CN2 deployment. You're not limited to the deployment described in this section nor are you limited to using the provided manifests. CN2 supports a wide range of deployments that are too numerous to cover in detail. Use the provided examples as a starting point to roll your own manifest for your specific situation.

1. Create the central cluster.

Follow the example procedure in "Create a Rancher RKE2 Cluster" on page 59 or you can use any other method. Create the cluster with no CNI plug-in.

Tailor the procedure with the desired number of server (control plane) and agent (worker) nodes accordingly.

2. Install CN2 on the central cluster.

   a. Apply the central cluster manifest. This manifest creates the namespaces and other resources required by the central cluster. It also creates the contrail-k8s-deployer deployment, which deploys CN2 and provides life cycle management for the CN2 components.

   ```
   user@central:~/contrail$ kubectl apply -f manifests/central_cluster_deployer_example.yaml
   ```

   b. Check that all pods are now up. This might take a few minutes.

   ```
   kubectl get pods -A -o wide
   ```

   You've now created the central cluster.

3. Follow "Attach a Workload Cluster" on page 50 to create and attach a distributed workload cluster to the central cluster.
4. Repeat step 3 for every workload cluster you want to create and attach.
5. (Optional) Run postflight checks. See "Run Preflight and Postflight Checks" on page 45.

NOTE: Run postflight checks from the central cluster only.

Install Contrail Tools

SUMMARY

Learn how to install tools that can help your CN2 installation go more smoothly.

IN THIS SECTION

- Install ContrailReadiness Controller | 30

Contrail tools are implemented within the ContrailReadiness controller framework. The controller runs the tools and gathers and presents the results asynchronously on demand.

You'll need to set up the ContrailReadiness controller framework before you can run any tools. After the controller comes up, follow the procedure for the tool you would like to run.

- "preflight" on page 45
- "postflight" on page 45
- "CN2 uninstall" on page 48

Install ContrailReadiness Controller

Use this procedure to install the ContrailReadiness controller. The ContrailReadiness controller is required before you can run any tools.

You can install the ContrailReadiness controller before or after you install CN2. Installing the controller before you install CN2 allows you to run preflight checks on the cluster.

1. Locate the **contrail-tools/contrail-readiness** directory from the downloaded CN2 Tools package.
2. If you haven't already done so, ensure you've populated the tools manifests with your repository login credentials. See "Configure Repository Credentials" on page 66 for one way to do this.
3. Apply the ContrailReadiness custom resource definitions.

   ```
   kubectl apply -f contrail-tools/contrail-readiness/crds
   ```

4. Create the ConfigMap from the deployer manifest that you plan to use or have used to install this cluster. Name the ConfigMap `deployer-yaml`.

   ```
   kubectl create configmap deployer-yaml --from-file=<path_to_deployer_manifest>
   ```

   where *<path_to_deployer_manifest>* is the full path to the deployer manifest that you want to apply or have applied.
5. Patch the ConfigMap with the registry information.

   ```
   kubectl patch configmap deployer-yaml --type merge -p '{"data":{"registry":"enterprise-
   hub.juniper.net/contrail-container-prod"}}'
   ```

6. Create the ContrailReadiness controller.

   ```
   kubectl apply -f contrail-tools/contrail-readiness/contrail-readiness-controller.yaml
   ```

Check that the controller has come up.

```
kubectl get pods -n contrail-readiness
```

Manifests

SUMMARY

We provide sample manifests to make your installation easier. You can download these manifests from the Juniper Networks software download site or from GitHub.

IN THIS SECTION

- Manifests in Release 23.3 | **31**
- Contrail Tools in Release 23.3 | **32**
- Contrail Analytics in Release 23.3 | **33**

Manifests in Release 23.3

The CN2 manifests for Rancher RKE2 are identical to the CN2 manifests for Upstream Kubernetes.

The CN2 Upstream Kubernetes manifests package is called **Deployment Manifests for K8s** and is available for download from the Juniper Networks software download site (https://support.juniper.net/support/downloads/?p=contrail-networking) or from github (https://github.com/Juniper/contrail-networking/tree/main/releases/23.3/k8s).

NOTE: The provided manifests might not be compatible between releases. Make sure you use the manifests for the release that you're running. In practice, this means that you should not modify the image tag in the supplied manifests.

If you're downloading from the Juniper Networks software download site, you'll need an account to download. If you don't have an account, contact your Juniper Networks sales representative to have one created for you.

The following table lists the single cluster manifests in that package.

Table 4: Single Cluster Manifests for Rancher RKE2 for Release 23.3

Manifests	Description
k8s/single_cluster/ single_cluster_deployer_example.yaml	Contains the manifests to install Contrail in a single cluster.

The following table lists the manifests that are specific to setting up a multi-cluster.

Table 5: Multi-Cluster Manifests for Rancher RKE2 for Release 23.3

Manifests	Description
k8s/multi-cluster/ central_cluster_deployer_example.yaml	Contrail deployer and necessary resources for the central cluster in a multi-cluster setup.
k8s/multi-cluster/ distributed_cluster_certmanager_example.yaml	Contrail cert-manager manifests for encrypting Contrail management and control plane communications.
k8s/multi-cluster/ distributed_cluster_deployer_example.yaml	Contrail deployer and necessary resources for distributed workload clusters in a multi-cluster setup.
k8s/multi-cluster/ distributed_cluster_vrouter_example.yaml	Contrail vRouter for the distributed workload clusters in a multi-cluster setup.

Contrail Tools in Release 23.3

IN THIS SECTION

- Contrail Tools | 33

Contrail Tools

The optional Contrail Tools package is called **Contrail Tools** and is available for download from the Juniper Networks software download https://support.juniper.net/support/downloads/?p=contrail-networking site. Contrail tools are compatible with CN2 within the same release only.

You'll need an account to download. If you don't have an account, contact your Juniper Networks sales representative to have one created for you.

The following table lists tools that we provide.

Table 6: Tools Manifests for Release 23.3

Tools	Description
contrail-tools/contrail-readiness/contrail-readiness-controller.yaml	The ContrailReadiness controller that runs preflight and postflight checks
contrail-tools/contrail-readiness/contrail-readiness-preflight.yaml	ContrailReadiness preflight custom resource
contrail-tools/contrail-readiness/contrail-readiness-postflight.yaml	ContrailReadiness postflight custom resource
contrail-tools/contrail-readiness/contrail-readiness-uninstall.yaml	ContrailReadiness uninstall custom resource
contrail-tools/contrail-readiness/crds	ContrailReadiness custom resource definitions for the supported tools
contrail-tools/kubectl-contrailstatus-*<release>*.tar	The kubectl contrailstatus plug-in
contrail-tools/cn2_debug_infra-*<release>*.tar	The CN2 debug utility
contrail-tools/uninstall.tar.gz	Deprecated

Contrail Analytics in Release 23.3

The optional Contrail Analytics package is called **Analytics Deployer** and is available for download from the Juniper Networks software download https://support.juniper.net/support/downloads/?p=contrail-networking site. Select the Contrail Analytics package from the same release page that you select the

Contrail Networking manifests. Contrail Analytics is compatible with Contrail Networking within the same release only.

You'll need an account to download. If you don't have an account, contact your Juniper Networks sales representative to have one created for you.

To install Contrail Analytics, see the *Install Contrail Analytics and the CN2 Web UI* section.

3

CHAPTER

Monitor

Overview | 36

Install Contrail Analytics and the CN2 Web UI | 36

Kubectl Contrailstatus | 39

Overview

You can monitor CN2 in the same way you monitor other Kubernetes components, using kubectl or other standard Kubernetes methods.

You can also install the optional Contrail Analytics package, which packages Prometheus, Grafana, Fluentd, and other popular open source software together with Contrail telemetry exporters to provide you with insight into the general health, performance, and traffic trends of the network. Included with Contrail Analytics is the CN2 Web UI, which you can use to monitor and configure CN2 components.

Additionally, we provide a kubectl plug-in that you can invoke to check the status of CN2 components from the command line. The contrailstatus plug-in allows you to query the CN2 configuration, control, and data plane components as well as BGP and XMPP relationships.

Install Contrail Analytics and the CN2 Web UI

Use this procedure to install Contail Analytics and the CN2 Web UI.

Contrail Analytics packages popular open source software such as Prometheus, Grafana, and Fluentd together with CN2 telemetry exporters to provide an industry-standard way for you to monitor and analyze your network and network infrastructure. Information collected includes logs, metrics, status' of various component, and flows.

Packaged with Contrail Analytics is the CN2 Web UI, which allows you to monitor and configure CN2 components.

When you install Contrail Analytics, all analytics components are preconfigured to work with each other.

You have the option of installing Contrail Analytics with a single instance of Prometheus or with HA Prometheus support. HA Prometheus for Contrail Analytics is a Tech Preview feature.

NOTE: We use Helm charts to install Contrail Analytics. Install Helm 3.0 or later on the host that you're using to install Contrail Analytics.

1. Locate the Contrail Analytics package that you downloaded.

   ```
   contrail-analytics-<version>.tgz
   ```

2. To install Contrail Analytics with a single instance of Prometheus:

   ```
   helm -n contrail-analytics install analytics contrail-analytics-<version>.tgz --create-namespace
   ```

   The `--create-namespace` option creates the contrail-analytics namespace. You can omit this option if your cluster already has the contrail-analytics namespace defined.

   Contrail Analytics is installed as a NodePort service. You can reach the service by specifying the IP address of any node running Contrail Analytics. By default, the port to use is 30443.
3. To install Contrail Analytics with HA Prometheus support (Tech Preview):

   NOTE: This feature is classified as a Juniper CN2 Technology Preview feature. These features are "as is" and are for voluntary use. Juniper Support will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features.

   For additional information, refer to the "Juniper CN2 Technology Previews (Tech Previews)" on page 68 or contact Juniper Support.

   a. Extract the **thanos-values.yaml** file from the Contrail Analytics package.

      ```
      tar --strip=1 -xzf contrail-analytics-<version>.tgz contrail-analytics/thanos-values.yaml
      ```

      Contrail Analytics uses Thanos to provide high availability for Prometheus. Thanos is a set of open source components that integrate seamlessly with Prometheus to provide a highly available metric system.

   b. Install Contrail Analytics (referencing the **thanos-values.yaml**) file.

      ```
      helm -n contrail-analytics install analytics contrail-analytics-<version>.tgz -f thanos-values.yaml --create-namespace
      ```

      The `--create-namespace` option creates the contrail-analytics namespace. You can omit this option if your cluster already has the contrail-analytics namespace defined.

      Contrail Analytics is installed as a NodePort service. You can reach the service by specifying the IP address of any node running Contrail Analytics. By default, the port to use is 30443.

4. Verify that the analytics components are installed and running.

```
helm -n contrail-analytics list
```

```
kubectl get pods -n contrail-analytics
```

5. After you install Contrail Analytics, you can access Grafana or the CN2 Web UI.

   To access Grafana, point your browser to https:// *<node-IP-address>*:30443/grafana/. Be sure to include the trailing /. The default Grafana administrator username/password is `admin/prom-operator`.

   To access the CN2 Web UI, point your browser to https:// *<node-IP-address>*:30443. The default CN2 Web UI username/password is `super/c0ntrail123`.

   NOTE: The CN2 Web UI is classified as a Juniper CN2 Technology Preview feature. These features are "as is" and are for voluntary use. Juniper Support will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features.

   For additional information, refer to the "Juniper CN2 Technology Previews (Tech Previews)" on page 68 or contact Juniper Support.

6. To uninstall Contrail Analytics:

```
helm -n contrail-analytics uninstall analytics
```

```
kubectl delete ns contrail-analytics
```

7. To upgrade Contrail Analytics:

```
helm -n contrail-analytics upgrade analytics contrail-analytics-<version>.tgz
```

   or (for upgrading HA)

```
helm -n contrail-analytics upgrade analytics contrail-analytics-<version>.tgz -f thanos-values.yaml
```

Kubectl Contrailstatus

IN THIS SECTION

- Syntax | 39
- Description | 39
- Options | 40
- Additional Information | 40
- Output Fields | 41
- Sample Output | 42
- Release Information | 43

Syntax

```
kubectl contrailstatus deployment --plane { config | control | data [--wide] }
kubectl contrailstatus resource { bgprouter [BGP | XMPP] | globalsystemconfig | routinginstance
| virtualnetwork [--wide] }
kubectl contrailstatus cresource all [detail]
kubectl contrailstatus configdump
kubectl contrailstatus --all [--wide]
kubectl contrailstatus version
```

Description

This command displays the status' of various CN2 components. You can display the status' of the Configuration plane components, the Control plane components, the Data plane components, and the BGP routers and other resources.

Options

kubectl contrailstatus deployment --plane config	Displays the status of the Configuration plane components: • contrail-k8s-apiserver • contrail-k8s-controller • contrail-k8s-kubemanager
kubectl contrailstatus deployment --plane control	Displays the status of the Control plane components: • contrail-control
kubectl contrailstatus deployment --plane data	Displays the status of the Data plane components: • contrail-vrouter-masters • contrail-vrouter-nodes
kubectl contrailstatus resource bgprouter	Displays the status' of the various BGP and XMPP neighbor relationships.
kubectl contrailstatus resource globalsystemconfig	Displays the status of the GlobalSystemConfig.
kubectl contrailstatus resource routinginstance	Displays the status' of the various RoutingInstances in CN2.
kubectl contrailstatus resource virtualnetwork	Displays the status' of the various VirtualNetworks in CN2.
kubectl contrailstatus cresource all	Displays all information about all resources (useful for displaying all information in a single command for debugging). If the `detail` option is used, the output is displayed in JSON format.
kubectl contrailstatus configdump	Lists the resources and their quantities.
kubectl contrailstatus --all	Displays the status' of the Configuration/Control/Data planes and the BGP and XMPP relationships.
kubectl contrailstatus version	Displays the versions of the various container images.

Additional Information

The `--wide` qualifier displays more information (if available) on the queried component.

Use the `--help` qualifier to display the help at any point in the command.

This command looks for the kubeconfig file in the default **~/.kube/config** location. You can't use the **kubectl --kubeconfig** option to specify the location of the kubeconfig file.

Output Fields

Table 7 on page 41 lists some of the output fields for the `kubectl contrailstatus` command.

Table 7: kubectl contrailstatus Output Fields

Field Name	Field Description
NAME	The name of the pod or resource.
STATUS	The status of the pod or resource.
NODE	The name of the node on which the pod is running.
IP	The (machine) IP address of the node on which the pod is running.
MESSAGE	Not used.
LOCAL BGPROUTER	The name of the node on which the local BGP router is running.
NEIGHBOR BGPROUTER	The name of the node on which the neighbor BGP router is running.
ENCODING	Whether this connection is XMPP or BGP.
STATE	The state of this connection.
POD	The name of the pod on which the local BGP router is running.

Sample Output

kubectl contrail-status --all

```
user@host> kubectl contrail-status --all
```

```
NAME(CONFIG)                                    STATUS  NODE    IP              MESSAGE
contrail-k8s-apiserver-6d79c8598d-8lfnm         ok      ocp1    172.16.0.11
contrail-k8s-apiserver-6d79c8598d-q7klk         ok      ocp3    172.16.0.13
contrail-k8s-apiserver-6d79c8598d-szdzf         ok      ocp2    172.16.0.12
contrail-k8s-controller-96964f568-csk2k         ok      ocp1    172.16.0.11
contrail-k8s-controller-96964f568-dshn6         ok      ocp3    172.16.0.13
contrail-k8s-controller-96964f568-hfrpl         ok      ocp2    172.16.0.12
contrail-k8s-kubemanager-79b577ff86-6v8qt       ok      ocp3    172.16.0.13
contrail-k8s-kubemanager-79b577ff86-cbh5n       ok      ocp1    172.16.0.11
contrail-k8s-kubemanager-79b577ff86-vmckw       ok      ocp2    172.16.0.12


NAME(CONTROL)           STATUS  NODE    IP              MESSAGE
contrail-control-0      ok      ocp1    172.16.0.11
contrail-control-1      ok      ocp2    172.16.0.12
contrail-control-2      ok      ocp3    172.16.0.13


LOCAL BGPROUTER NEIGHBOR BGPROUTER      ENCODING        STATE           POD
ocp1            ocp2                    BGP             Established ok  contrail-control-0
ocp1            ocp3                    BGP             Established ok  contrail-control-0
ocp1            ocp1                    XMPP            Established ok  contrail-control-0
ocp1            ocp2                    XMPP            Established ok  contrail-control-0
ocp1            ocp3                    XMPP            Established ok  contrail-control-0
ocp1            ocp4                    XMPP            Established ok  contrail-control-0
ocp1            ocp5                    XMPP            Established ok  contrail-control-0
ocp2            ocp3                    BGP             Established ok  contrail-control-1
ocp2            ocp1                    BGP             Established ok  contrail-control-1
ocp2            ocp2                    XMPP            Established ok  contrail-control-1
ocp2            ocp3                    XMPP            Established ok  contrail-control-1
ocp2            ocp4                    XMPP            Established ok  contrail-control-1
ocp2            ocp5                    XMPP            Established ok  contrail-control-1
```

```
ocp3            ocp1                   BGP             Established ok   contrail-control-2
ocp3            ocp2                   BGP             Established ok   contrail-control-2
ocp3            ocp1                   XMPP            Established ok   contrail-control-2


NAME(DATA)                        STATUS  NODE    IP              MESSAGE
contrail-vrouter-masters-dspzb  ok      ocp3    172.16.0.13
contrail-vrouter-masters-ks249  ok      ocp2    172.16.0.12
contrail-vrouter-masters-tn6jz  ok      ocp1    172.16.0.11
contrail-vrouter-nodes-mjwt2    ok      ocp4    172.16.0.14
contrail-vrouter-nodes-rp5np    ok      ocp5    172.16.0.15
```

Release Information

Table 8: Summary of Changes

Release	Changes
22.1	Initial release.
22.4	Updated `version` command to include image versions. Added `cresource` and `configdump` commands.

4

CHAPTER

Manage

Manage Single Cluster CN2 | 45

Manage Multi-Cluster CN2 | 49

Manage Single Cluster CN2

SUMMARY

Learn how to perform life cycle management tasks in a single cluster installation or within a specific cluster in a multi-cluster installation.

IN THIS SECTION

- Overview | 45
- Run Preflight and Postflight Checks | 45
- Upgrade CN2 | 47
- Uninstall CN2 | 48

Overview

The way that you manage a Kubernetes cluster does not change when CN2 is the CNI plug-in. Once CN2 is installed, CN2 components work seamlessly with Kubernetes components to provide the networking infrastructure.

The Contrail controller is constantly watching and reacting to cluster events as they occur. When you add a new node, the Contrail data plane components are automatically deployed. When you delete a node, the Contrail controller automatically deletes networking resources associated with that node. CN2 works seamlessly with kubectl and other tools such as Prometheus and Grafana.

In addition to standard Kubernetes management tools, you can use tools and procedures that are specific to CN2. This section covers these tools and procedures.

Run Preflight and Postflight Checks

Use this procedure to run preflight or postflight checks on all cluster nodes.

Preflight checks allow you to verify that your cluster nodes can support CN2. The checks test for resource capacity, kernel compability, network reachability, and other infrastructure requirements. You typically run preflight checks prior to installing CN2, but you can run these checks after installing CN2 as well.

Postflight checks allow you to verify that your CN2 installation is working properly. The checks test for status, pod-to-pod communication, API server reachability, and other basic functions. You run postflight checks after installing CN2.

Before you can run this procedure, ensure you've installed the ContrailReadiness controller. The ContrailReadiness controller provides the framework for preflight and postflight checks.

1. Locate the **contrail-tools/contrail-readiness** directory from the downloaded CN2 Tools package.
2. If you haven't already done so, ensure you've populated the manifests with your repository login credentials. See "Configure Repository Credentials" on page 66 for one way to do this.
3. To run the preflight checks:

   ```
   kubectl apply -f contrail-tools/contrail-readiness/contrail-readiness-preflight.yaml
   ```

   You typically run preflight checks after you create the cluster but before you install CN2.

   NOTE: In a multi-cluster deployment, run preflight checks from the central cluster only.

4. To run the postflight checks:

   ```
   kubectl apply -f contrail-tools/contrail-readiness/contrail-readiness-postflight.yaml
   ```

   You run postflight checks after you install CN2.

   NOTE: In a multi-cluster deployment, run postflight checks from the central cluster only.

5. Read the preflight and postflight check results as applicable.

   ```
   kubectl get contrailreadiness preflight -o yaml
   ```

   ```
   kubectl get contrailreadinesstask preflight-kernel -o yaml
   ```

   ```
   kubectl get contrailreadiness postflight -o yaml
   ```

   ```
   kubectl get contrailreadinesstask postflight-contrailresources -o yaml
   ```

   Address any errors before proceeding.

NOTE: The preflight and postflight checks do not automatically rerun after you've fixed any errors. The output will continue to show errors even after you've fixed them.

Upgrade CN2

Use this procedure to upgrade CN2.

The Contrail controller consists of Deployments and StatefulSets, which are configured for rolling updates. During the upgrade, the pods in each Deployment and StatefulSet are upgraded one at a time. The remaining pods in that Deployment or StatefulSet remain operational. This enables Contrail controller upgrades to be hitless.

The Contrail data plane consists of a DaemonSet with a single vRouter pod. During the upgrade procedure, this single pod is taken down and upgraded. Because of this, Contrail data plane upgrades are not hitless. If desired, migrate traffic off of the node being upgraded prior to performing the upgrade.

You upgrade CN2 software by porting the contents of your existing manifests to the new manifests, and then applying the new manifests. All CN2 manifests must reference the same software version.

NOTE: Before you upgrade, check to make sure that each node has at least one allocatable pod available. The upgrade procedure temporarily allocates an additional pod, which means that your node cannot be running at maximum pod capacity when you perform the upgrade. You can check pod capacity on a node by using the `kubectl describe node` command.

1. Download the manifests for the new release.
2. Locate the (old) manifest(s) that you used to create the existing CN2 installation. In this procedure, we assume it's **single_cluster_deployer_example.yaml**.
3. Port over any changes from the old manifest(s) to the new manifest(s).

   The new manifests can contain constructs that are specific to the new release. Identify all changes that you've made to the old manifests and copy them over to the new manifests. This includes repository credentials, network configuration changes, and other customizations.

   NOTE: If you have a large number of nodes, use node selectors to group your upgrades to a more manageable number.

4. Upgrade CN2.

   ```
   kubectl apply -f manifests/single_cluster_deployer_example.yaml
   ```

   The pods in each Deployment and Stateful set will upgrade one at a time. The vRouter DaemonSet will go down and come back up.
5. Use standard kubectl commands to check on the upgrade.

   Check the status of the nodes.

   ```
   kubectl get nodes
   ```

   Check the status of the pods.

   ```
   kubectl get pods -A -o wide
   ```

   If some pods remain down, debug the installation as you normally do. Use the `kubectl describe` command to see why a pod is not coming up. A common error is a network or firewall issue preventing the node from reaching the Juniper Networks repository.

Uninstall CN2

Use this procedure to uninstall CN2.

This tool removes the following:

- contrail namespace and resources that belong to that namespace
- contrail-system namespace and resources that belong to that namespace
- contrail-deploy namespace and resources that belong to that namespace
- default-global-vrouter-config and default-global-system-config

Before you can run this procedure, ensure you've installed the ContrailReadiness controller. The ContrailReadiness controller provides the framework for the uninstall task.

NOTE: Since there are interdependencies between CN2 components, don't try to delete CN2 components individually. The provided tool uninstalls CN2 components gracefully and in the proper sequence.

1. Locate the **contrail-tools/contrail-readiness** directory from the downloaded CN2 Tools package.
2. If you haven't already done so, ensure you've populated the manifests with your repository login credentials. See "Configure Repository Credentials" on page 66 for one way to do this.
3. If you've installed Contrail Analytics, uninstall it now. The uninstall script does not uninstall resources in namespaces other than those listed above. To uninstall Contrail Analytics, see the *Install Contrail Analytics and the CN2 Web UI* section.
4. Delete any other resources and namespaces (for example, overlay networks) that you created after you installed CN2.
5. Uninstall CN2.

```
kubectl apply -f contrail-tools/contrail-readiness/contrail-readiness-uninstall.yaml
```

6. Query the uninstall results.

```
kubectl get contrailreadiness contrail-uninstall -o yaml
```

7. Finally, delete the contrail-readiness namespace.

```
kubectl delete ns contrail-readiness
```

Manage Multi-Cluster CN2

SUMMARY

Learn how to perform life cycle management tasks specific to a multi-cluster installation.

IN THIS SECTION

- Attach a Workload Cluster | 50
- Detach a Workload Cluster | 55

- Uninstall CN2 | 56

This section covers tasks that are specific to a multi-cluster installation. If you want to perform management tasks in a specific cluster (such as adding and removing nodes within a cluster and upgrading a cluster) within the multi-cluster installation, then see "Manage Single Cluster CN2" on page 45.

Attach a Workload Cluster

Use this procedure to create and attach a distributed workload cluster (running a kernel mode data plane) to the central cluster. If you want to run a DPDK data plane, then extrapolate from the DPDK examples in this document.

The manifests that you will use in this example procedure are **k8s/multi-cluster/distributed_cluster_deployer_example.yaml** and **k8s/multi-cluster/distributed_cluster_vrouter_example.yaml**. The procedure assumes that you've placed these manifests into a **manifests** directory.

1. Prepare the distributed workload cluster as described in "Before You Install" on page 18.
2. Create the distributed workload cluster.

   Create a fresh cluster. You can follow the procedure in "Create a Rancher RKE2 Cluster" on page 59 or use your own methods to create a cluster. The cluster should have the following characteristics:

   - Cluster has no CNI plug-in.
   - In a multi-cluster setup, you must configure different pod and service subnets on each cluster. These subnets must be unique within the entire multi-cluster.
   - In a multi-cluster setup, you must configure different node names for nodes in each cluster. Node names must be unique across the entire multi-cluster.
3. Install CN2 components on the distributed workload cluster.

   a. On the workload cluster, copy the kubeconfig from the central cluster. We'll call this **central-cluster-kubeconfig** here.

   ```
   scp user@central:~/.kube/config central-cluster-kubeconfig
   ```

b. On the workload cluster, create the contrail-deploy namespace.

```
kubectl create ns contrail-deploy
```

c. On the workload cluster, create a Kubernetes secret from the central cluster kubeconfig and name the secret **central-kubeconfig**.

NOTE: You must name the secret **central-kubeconfig**.

```
kubectl create secret generic central-kubeconfig -n contrail-deploy --from-
file=kubeconfig=/root/contrail/central-cluster-kubeconfig
```

NOTE: You must specify the absolute path to the **central-cluster-kubeconfig** file.

d. On the workload cluster, apply the deployer manifest. The deployer provides life cycle management for the CN2 components.

This manifest includes a reference to the **central-kubeconfig** secret you created in the previous substep.

```
kubectl apply -f manifests/distributed_cluster_deployer_example.yaml
```

e. Check that the contrail-deployer has come up. This may take a few minutes.

```
kubectl get pods -n contrail-deploy
```

```
NAME                                    READY   STATUS    RESTARTS   AGE
contrail-k8s-deployer-6458859585-xhwx6   1/1     Running   0          6m
```

f. On the workload cluster, apply the cert-manager manifest. The cert-manager provides encryption for all management and control plane connections.

```
kubectl apply -f manifests/distributed_cluster_certmanager_example.yaml
```

4. On the central cluster, attach the new workload cluster by creating a kubemanager for the new workload cluster.

   a. On the central cluster, copy the kubeconfig from the distributed workload cluster. We'll call this **workload-cluster-kubeconfig** here.

   ```
   scp user@workload:~/.kube/config workload-cluster-kubeconfig
   ```

   b. On the central cluster, create a Kubernetes secret from the distributed workload cluster kubeconfig and choose a meaningful name for the secret (for example, **workload-kubeconfig**).

   ```
   kubectl create secret generic workload-kubeconfig -n contrail --from-file=kubeconfig=/root/
   contrail/workload-cluster-kubeconfig
   ```

   c. Create the kubemanager manifest with the following content. Choose a meaningful name for the manifest (for example, **kubemanager-cluster1.yaml**).

   ```
   apiVersion: configplane.juniper.net/v1alpha1
   kind: Kubemanager
   metadata:
     name: <CR name>
     namespace: contrail
   spec:
     common:
       containers:
       - image: <contrail-image-repository>
         name: contrail-k8s-kubemanager
     podV4Subnet: <pod-v4-subnet-of-remote-cluster>
     serviceV4Subnet: <service-v4-subnet-of-remote-cluster>
     podV6Subnet: <pod-v6-subnet-of-remote-cluster>
     serviceV6Subnet: <service-v6-subnet-of-remote-cluster>
     clusterName: <worker-cluster-name>
     kubeconfigSecretName: <secret-name>
     enableNad: <true/false>
     listenerPort: <listener-port>
     constantRouteTargetNumber: <rt-number>
   ```

   Table 9 on page 53 explains the parameters that you need to set.

Table 9: Kubemanager CRD

Parameter	Meaning	Example
name	The name of the custom resource.	kubemanager-cluster1
image	The repository where you pull images	enterprise-hub.juniper.net/contrail-container-prod/contrail-k8s-kubemanager:23.3.0.180
podV4Subnet	The IPv4 pod subnet that you configured earlier for the distributed workload cluster. This subnet must be unique within the entire multi-cluster.	10.234.64.0
serviceV4Subnet	The IPv4 service subnet that you configured earlier for the distributed workload cluster. This subnet must be unique within the entire multi-cluster.	10.234.0.0/18
podV6Subnet	The IPv6 pod subnet that you configured earlier for the distributed workload cluster. This subnet must be unique within the entire multi-cluster.	fd85:ee78:d8a6:8608::1:0000/112
serviceV6Subnet	The IPv6 service subnet that you configured earlier for the distributed workload cluster. This subnet must be unique within the entire multi-cluster.	fd85:ee78:d8a6:8608::1000/116
clusterName	The name of the workload cluster.	workload-cluster
kubeconfigSecretName	The name of the secret containing the workload cluster kubeconfig token.	workload-kubeconfig

Table 9: Kubemanager CRD *(Continued)*

Parameter	Meaning	Example
enableNad	True or false (to enable network address definition or not)	true
listenerPort	The port that the Contrail controller listens on for communications with this workload cluster. Set the port for the first workload cluster to 19446 and increment by 1 for each subsequent workload cluster.	19446
constantRouteTargetNumber	The route target for this workload cluster. Set the route target for the first workload cluster to 7699 and increment by 100 for each subsequent workload cluster.	7699

d. On the central cluster, apply the kubemanager manifest you just created.

```
kubectl apply -f manifests/kubemanager-cluster1.yaml
```

e. On the central cluster, verify that you can see the workload cluster's namespaces.

```
kubectl get ns
```

The namespaces are in the following format: *<kubemanager-name>*-*<workload-cluster-name>*-*<namespace>*. For example:

```
kubemanager-workload1-workload-cluster-contrail
```

5. Finally, on the workload cluster, install the vRouter.

```
kubectl apply -f manifests/distributed_cluster_vrouter_example.yaml
```

You've now created and attached a distributed workload cluster to the central cluster.

Detach a Workload Cluster

Use this procedure to detach a distributed workload cluster from the central cluster.

1. On the central cluster, delete the associated kubemanager.

```
kubectl delete kubemanager -n contrail <kubemanager-name>
```

2. On the distributed workload cluster that you're deleting, delete the vRouters.

```
kubectl delete vrouter -n contrail contrail-vrouter-masters
```

```
kubectl delete vrouter -n contrail contrail-vrouter-nodes
```

3. On the central cluster, delete the namespaces of the distributed workload cluster.

   List the namespaces and delete all namespaces associated with the distributed workload cluster.

```
kubectl get ns
```

```
kubectl delete ns <kubemanager-name>-<workload-cluster-name>-<namespace>
```

4. On the central cluster, delete the secret associated with the workload cluster that you're detaching.

```
kubectl delete secret <secret-name> -n contrail
```

   where *<secret-name>* is the secret you created when you attached the workload cluster to the central cluster earlier. In our example, we called it **workload-kubeconfig**.

Uninstall CN2

Use this procedure to uninstall CN2 from the central cluster and the workload clusters.

1. Follow the steps in "Detach a Workload Cluster" on page 55 to detach the workload cluster that you want to delete.
2. Uninstall CN2 in the workload cluster.
   a. Follow the steps in "Uninstall CN2" on page 48 to uninstall CN2 in the workload cluster.

      Ignore any NotFound errors because those resources have already been deleted.

   b. On the workload cluster, verify that all resources related to default pod network namespace of the distributed workload cluster are gone, and verify that all resources related to the contrail namespace are gone.

      ```
      kubectl api-resources --verbs=list --namespaced -o name | xargs -n 1 kubectl get --show-
      kind --ignore-not-found -n <default pod network namespace>
      ```

      ```
      kubectl api-resources --verbs=list --namespaced -o name | xargs -n 1 kubectl get --show-
      kind --ignore-not-found -n contrail
      ```

   If all you want to do is to uninstall CN2 from the workload cluster, then you're done. If you want to also uninstall CN2 from the central cluster, then proceed to the next step.
3. Uninstall CN2 in the central cluster.
   a. Follow the steps in "Uninstall CN2" on page 48 to uninstall CN2 in the central cluster.

      Ignore any NotFound errors because those resources have already been deleted.

b. On the central cluster, verify that all resources related to the contrail, contrail-system, and contrail-deploy namespaces are deleted.

```
kubectl api-resources --verbs=list --namespaced -o name | xargs -n 1 kubectl get --show-
kind --ignore-not-found -n contrail
```

```
kubectl api-resources --verbs=list --namespaced -o name | xargs -n 1 kubectl get --show-
kind --ignore-not-found -n contrail-system
```

```
kubectl api-resources --verbs=list --namespaced -o name | xargs -n 1 kubectl get --show-
kind --ignore-not-found -n contrail-deploy
```

5

CHAPTER

Appendix

Create a Rancher RKE2 Cluster | 59

Configure Repository Credentials | 66

Prepare a Cluster Node for DPDK | 67

Juniper CN2 Technology Previews (Tech Previews) | 68

Create a Rancher RKE2 Cluster

SUMMARY

See examples on how to create a Rancher RKE2 cluster.

IN THIS SECTION

- Configure a Server Node | 59
- Configure an Agent Node | 63

Use this example procedure to create a Rancher RKE2 cluster.

This procedure configures a server node and two agent nodes. In Rancher, a server node functions as a control plane node, and an agent node functions as a worker node. Tailor this procedure to the number of nodes in your cluster.

We provide this example procedure purely for informational purposes.

For more information about creating an RKE2 cluster, see the official Rancher documentation: https://docs.rke2.io/install/quickstart.

Before you start, make sure you've brought up the servers or VMs that you plan to use for the cluster nodes.

NOTE: The command line examples below don't always show absolute directory paths. We leave it to you to apply these commands within your directory structure.

Configure a Server Node

Use this procedure to configure a server node.

A server node functions as a control plane node in RKE2. The server node used in our single cluster example is an Ubuntu host reachable at IP address 172.16.0.11.

1. From your local computer, SSH into the server node as the root user.

2. Create a `config.yaml` file at `/etc/rancher/rke2` with the following content.

```
cni:
  - none
```

3. Install, enable, and start the `rke2-server` service.

   a. Download the RKE2 installation script:

   ```
   curl -sfL https://get.rke2.io -o install.sh
   ```

   b. Make the installation script executable.

   ```
   chmod +x install.sh
   ```

   c. Set the installation variables to point to the desired release.

   ```
   INSTALL_RKE2_CHANNEL=latest
   ```

   ```
   INSTALL_RKE2_CHANNEL_URL=https://github.com/rancher/rke2/releases
   ```

   d. Run the installation script.

   ```
   ./install.sh
   ```

   This script installs the `rke2-server` service.

   e. Enable and start the `rke2-server` service.

   ```
   systemctl enable rke2-server.service
   ```

   ```
   systemctl start rke2-server.service
   ```

f. Verify the status of the `rke2-server` service.

```
systemctl status rke2-server
```

```
● rke2-server.service - Rancher Kubernetes Engine v2 (server)
     Loaded: loaded (/usr/local/lib/systemd/system/rke2-server.service; enabled; vendor
preset: enabled)
     Active: active (running) since Tue 2023-06-06 17:37:38 UTC; 1 day 22h ago
```

4. Copy the kubeconfig file into the default kubeconfig directory.

```
mkdir ~/.kube
```

```
cp /etc/rancher/rke2/rke2.yaml ~/.kube/config
```

5. Copy kubectl into your default path. For convenience, Rancher provides the kubectl binary at the location shown.

```
cp /var/lib/rancher/rke2/bin/kubectl /usr/local/bin
```

6. Show the status of the pods.

```
kubectl get pods -A -o wide
```

```
NAMESPACE     NAME                                                  READY   STATUS      RESTARTS   AGE
IP            NODE      NOMINATED NODE   READINESS GATES
kube-system   cloud-controller-manager-rke2-s1                      1/1     Running     0          11m
172.16.0.11   rke2-s1   <none>           <none>
kube-system   etcd-rke2-s1                                          1/1     Running     0          11m
172.16.0.11   rke2-s1   <none>           <none>
kube-system   helm-install-rke2-coredns-trcjf                       0/1     Completed   0          10m
172.16.0.11   rke2-s1   <none>           <none>
kube-system   helm-install-rke2-ingress-nginx-jghfq                 0/1     Pending     0          10m
<none>        <none>    <none>           <none>
kube-system   helm-install-rke2-metrics-server-nvcp6                0/1     Pending     0          10m
<none>        <none>    <none>           <none>
```

```
kube-system   helm-install-rke2-snapshot-controller-9gbf7            0/1     Pending     0          10m
<none>        <none>    <none>           <none>
kube-system   helm-install-rke2-snapshot-controller-crd-l5wnn        0/1     Pending     0          10m
<none>        <none>    <none>           <none>
kube-system   helm-install-rke2-snapshot-validation-webhook-kqmnp    0/1     Pending     0          10m
<none>        <none>    <none>           <none>
kube-system   kube-apiserver-rke2-s1                                 1/1     Running     0          11m
172.16.0.11   rke2-s1   <none>           <none>
kube-system   kube-controller-manager-rke2-s1                        1/1     Running     0          11m
172.16.0.11   rke2-s1   <none>           <none>
kube-system   kube-proxy-rke2-s1                                     1/1     Running     0          11m
172.16.0.11   rke2-s1   <none>           <none>
kube-system   kube-scheduler-rke2-s1                                 1/1     Running     0          11m
172.16.0.11   rke2-s1   <none>           <none>
kube-system   rke2-coredns-rke2-coredns-6b9548f79f-snzl4             0/1     Pending     0          10m
<none>        <none>    <none>           <none>
kube-system   rke2-coredns-rke2-coredns-autoscaler-57647bc7cf-dc2gx  0/1     Pending     0          10m
<none>        <none>    <none>           <none>
```

Some pods are not running because CN2 is not yet installed.

7. Download and install the CNI plugin.

   a. Create the following directory for the CNI plugin.

   ```
   mkdir -p /opt/cni/bin
   ```

   b. Download the CNI plugin.

   ```
   cd /opt/cni/bin/
   ```

   ```
   wget 'https://github.com/containernetworking/plugins/releases/download/v1.1.1/cni-plugins-
   linux-amd64-v1.1.1.tgz'
   ```

   c. Untar and gunzip the .tgz file.

   ```
   tar -xzvf cni-plugins-linux-amd64-v1.1.1.tgz
   ```

Configure an Agent Node

Use this procedure to configure an agent node.

An agent node functions as a worker node in RKE2. The agent nodes used in our single cluster example are Ubuntu hosts reachable at IP addresses 172.16.0.12 and 172.16.0.13.

NOTE: Repeat these steps for the desired amount of agent nodes.

1. SSH into the agent node as the root user.
2. Create a `config.yaml` file in the `/etc/rancher/rke2` directory with the following content:

   ```
   server: https://<server_node_IP>:9345
   token: <server_node_token>
   ```

   The `server_node_IP` is the IP address of the server (control plane) node.

   The `server_node_token` is the token found in `/var/lib/rancher/rke2/server/node-token` on the server node.
3. Install, enable, and start the `rke2-agent` service.
   a. Download the RKE2 installation script:

      ```
      curl -sfL https://get.rke2.io -o install.sh
      ```

   b. Make the installation script executable.

      ```
      chmod +x install.sh
      ```

   c. Set the installation type.

      ```
      INSTALL_RKE2_TYPE="agent"
      ```

   d. Run the installation script.

      ```
      ./install.sh
      ```

      This script installs the `rke2-agent` service.

a. Enable and start the `rke2-agent` service.

```
systemctl enable rke2-agent.service
```

```
systemctl start rke2-agent.service
```

b. Verify the status of the `rke2-agent` service.

```
systemctl status rke2-agent
```

```
● rke2-agent.service - Rancher Kubernetes Engine v2 (agent)
     Loaded: loaded (/usr/local/lib/systemd/system/rke2-agent.service; enabled; vendor
preset: enabled)
     Active: active (running) since Mon 2023-06-12 17:36:21 UTC; 3min 6s ago
```

4. Show the status of the pods.

   Here's an example of the output when one agent node is up. Remember to issue this command from the server node.

```
kubectl get pods -A -o wide
```

```
NAMESPACE     NAME                                                   READY   STATUS      RESTARTS   AGE
IP            NODE      NOMINATED NODE   READINESS GATES
kube-system   cloud-controller-manager-rke2-s1                       1/1     Running     0          21m
172.16.0.11   rke2-s1   <none>           <none>
kube-system   etcd-rke2-s1                                           1/1     Running     0          21m
172.16.0.11   rke2-s1   <none>           <none>
kube-system   helm-install-rke2-coredns-trcjf                        0/1     Completed   0          20m
172.16.0.11   rke2-s1   <none>           <none>
kube-system   helm-install-rke2-ingress-nginx-jghfq                  0/1     Pending     0          20m
<none>        <none>    <none>           <none>
kube-system   helm-install-rke2-metrics-server-nvcp6                 0/1     Pending     0          20m
<none>        <none>    <none>           <none>
kube-system   helm-install-rke2-snapshot-controller-9gbf7            0/1     Pending     0          20m
<none>        <none>    <none>           <none>
kube-system   helm-install-rke2-snapshot-controller-crd-l5wnn        0/1     Pending     0          20m
```

```
<none>         <none>     <none>           <none>
kube-system    helm-install-rke2-snapshot-validation-webhook-kqmnp     0/1     Pending     0          20m
<none>         <none>     <none>           <none>
kube-system    kube-apiserver-rke2-s1                                  1/1     Running     0          21m
172.16.0.11    rke2-s1    <none>           <none>
kube-system    kube-controller-manager-rke2-s1                         1/1     Running     0          21m
172.16.0.11    rke2-s1    <none>           <none>
kube-system    kube-proxy-rke2-a1                                      1/1     Running     0          10m
172.16.0.12    rke2-a1    <none>           <none>
kube-system    kube-proxy-rke2-s1                                      1/1     Running     0          21m
172.16.0.11    rke2-s1    <none>           <none>
kube-system    kube-scheduler-rke2-s1                                  1/1     Running     0          21m
172.16.0.11    rke2-s1    <none>           <none>
kube-system    rke2-coredns-rke2-coredns-6b9548f79f-snzl4              0/1     Pending     0          20m
<none>         <none>     <none>           <none>
kube-system    rke2-coredns-rke2-coredns-autoscaler-57647bc7cf-dc2gx   0/1     Pending     0          20m
<none>         <none>     <none>           <none>
```

5. Download and install the CNI plugin.

   a. Create the following directory for the CNI plugin.

   ```
   mkdir -p /opt/cni/bin
   ```

   b. Download the CNI plugin.

   ```
   cd /opt/cni/bin/
   ```

   ```
   wget 'https://github.com/containernetworking/plugins/releases/download/v1.1.1/cni-plugins-
   linux-amd64-v1.1.1.tgz'
   ```

   c. Untar and gunzip the .tgz file.

   ```
   tar -xzvf cni-plugins-linux-amd64-v1.1.1.tgz
   ```

Configure Repository Credentials

Use this procedure to configure your repository login credentials in your manifests.

1. Install docker if you don't already have docker installed.
2. Log in to the Juniper Networks repository where you pull the container images.

   ```
   docker login enterprise-hub.juniper.net
   ```

   Enter your login credentials when prompted.

   Once you've logged in, your credentials are automatically stored in **~/.docker/config.json**. (If you installed docker using snap, then the credentials are stored in the **~/snap/docker** directory hierarchy.)
3. Encode your credentials in base64 and store the resulting string.

   ```
   ENCODED_CREDS=$(base64 -w 0 config.json)
   ```

   Take a look at the encoded credentials.

   ```
   echo $ENCODED_CREDS
   ```
4. Replace the credentials placeholder in the manifests with the encoded credentials.

   The manifests have a `<base64-encoded-credential>` credentials placeholder. Simply replace the placeholder with the encoded credentials in all manifests.

   ```
   sed -i s/'<base64-encoded-credential>'/$ENCODED_CREDS/ *.yaml
   ```

   Double check by searching for the encoded credentials in the manifests.

   ```
   grep $ENCODED_CREDS *.yaml
   ```

   You should see the encoded credentials in the manifests.

RELATED DOCUMENTATION

https://www.juniper.net/documentation/en_US/day-one-books/topics/concept/secrets.html

Prepare a Cluster Node for DPDK

Use this example procedure to prepare a cluster node for DPDK.

We provide this example procedure purely for informational purposes. See DPDK documentation (https://core.dpdk.org/doc/) for the official procedure.

1. Enable Intel VT-d and SR-IOV in the BIOS.
2. Configure huge pages and enable IOMMU by editing **/etc/default/grub**. For example (for a huge page size of 1 GB):

   ```
   GRUB_CMDLINE_LINUX_DEFAULT="default_hugepagesz=1G hugepagesz=1G hugepages=16 intel_iommu=on
   iommu=pt”
   ```

   Update grub and reboot:

   ```
   sudo update-grub
   ```

   ```
   sudo reboot
   ```

3. Load the poll mode driver (PMD) kernel module according to the capabilities of your NIC.

   - If your NIC supports VFIO, then you might not need to do anything explicitly. Recent kernels include VFIO compiled in. If you need to load it:

     ```
     sudo modprobe vfio-pci
     ```

   - If your NIC only supports UIO:

     ```
     sudo modprobe uio_pci_generic
     ```

     NOTE: Some kernel versions do not automatically include uio_pci_generic. If you see a `Module uio_pci_generic not found` error, then install the module first (for example: `sudo apt install linux-modules-extra-5.4.0-`*`xx`*`-generic`) before loading the module.

Juniper CN2 Technology Previews (Tech Previews)

Tech Previews enable you to test functionality and provide feedback during the development process of innovations that are not final production features. The goal of a Tech Preview is for the feature to gain wider exposure and potential full support in a future release. Customers are encouraged to provide feedback and functionality suggestions for a Technology Preview feature before it becomes fully supported.

Tech Previews may not be functionally complete, may have functional alterations in future releases, or may get dropped under changing markets or unexpected conditions, at Juniper's sole discretion. Juniper recommends that you use Tech Preview features in non-production environments only.

Juniper considers feedback to add and improve future iterations of the general availability of the innovations. Your feedback does not assert any intellectual property claim, and Juniper may implement your feedback without violating your or any other party's rights.

These features are "as is" and voluntary use. Juniper Support will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features. Certain features may have reduced or modified security, accessibility, availability, and reliability standards relative to General Availability software. Tech Preview features are not eligible for P1/P2 JTAC cases, and should not be subject to existing SLAs or service agreements.

For additional details, please contact Juniper Support or your local account team.