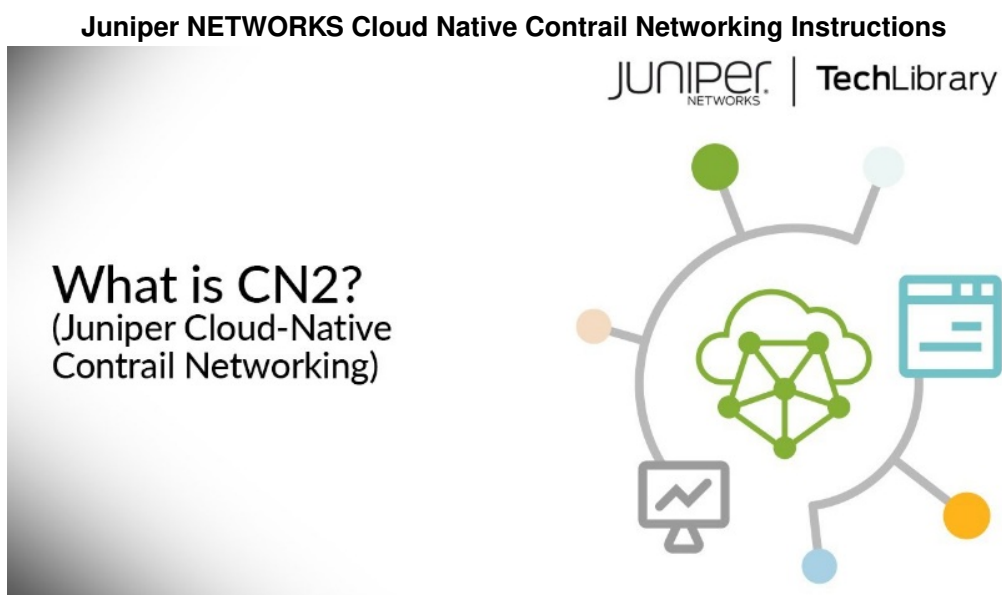




Juniper NETWORKS Cloud Native Contrail Networking Instructions

[Home](#) » [JUNIPER NETWORKS](#) » Juniper NETWORKS Cloud Native Contrail Networking Instructions 



Contents

- 1 Introduction
- 2 Cloud-Native Contrail Networking Overview
- 3 Benefits of Cloud-Native Contrail Networking
- 4 Terminology
- 5 CN2 Components
- 6 Deployment Models
- 7 Single Cluster Deployment
- 8 Multi-Cluster Deployment
- 9 System Requirements
- 10 Install
- 11 Overview
- 12 Benefits of Upstream Kubernetes with Contrail
- 13 Before You Install
- 14 Install Single Cluster Shared Network CN2
- 15 Install Single Cluster Shared Network CN2 Running DPDK Data Plane
- 16 Install Single Cluster Multi-Network CN2
- 17 Install Single Cluster Multi-Network CN2 Running DPDK Data Plane
- 18 Install Multi-Cluster Shared Network CN2
- 19 Install Multi-Cluster Shared Network CN2
- 20 Install Contrail Tools
- 21 Install ContrailReadiness Controller
- 22 Manifests
- 23 Manifests in Release 23.4
- 24 Contrail Tools
- 25 Contrail Analytics in Release 23.4
- 26 Monitor
- 27 Overview
- 28 Install Contrail Analytics and the CN2 Web UI
- 29 Documents / Resources
 - 29.1 References

Introduction

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 23.4, Cloud-Native Contrail Networking is supported on the following:

- (Upstream) Kubernetes
- Red Hat Open shift
- 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 multitenancy 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 Granma
- Support for both CRI-O and container 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.
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
Co nfi gur ati on Pla ne 1	contrail-k8s-apiserver	Control Plane Node	This pod is an aggregated API server th at is the entry point for managing all Co ntrail resources. It is registered with the regular kube EPiServer as an API Servi ce. The regular kube- EPiServer forwar ds all network-related requests to the c ontrail-k8s-apiserver for handling. Ther e is one contrail-k8s-apiserver pod per Kubernetes control plane node.
	contrail-k8s-controller	Control Plane Node	This pod performs the Kubernetes contr ol loop function to reconcile networking resources. It constantly monitors netwo rking resources to make sure the actual state of a resource matches its intended state. There is one contrail-k8 s-controller pod per Kubernetes control plane node.
	contrail-k8s- kubemanag er	Control Plane Node	This pod is the interface between Kubernetes resources and Contrail res ources. It watches the kube- apiserver f or changes to regular Kubernetes resou rces such as service and namespace a nd acts on any changes that affect the networking resources.In a single- cluster deployment, there is one contrai l-k8s-kubemanager pod per Kubernetes control plane node.In a multi-cluster de ployment, there is additionally one contr ail-k8s- kubemanager pod for every dist ributed 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-api server for anything affecting the network control plane and then communicates with its BGP peers and/or router agents (over XMPP) as appropriate. There is one contrail-control pod per Kubernetes control plane node.
Data Plane	contrail-vrouter-nodes	Worker Node	<p>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:</p> <ul style="list-style-type: none"> 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 <p>The vRouter provides the packet send and receive function for the co-located pods and workloads. It provides the CNF plug-in functionality.</p>
	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.

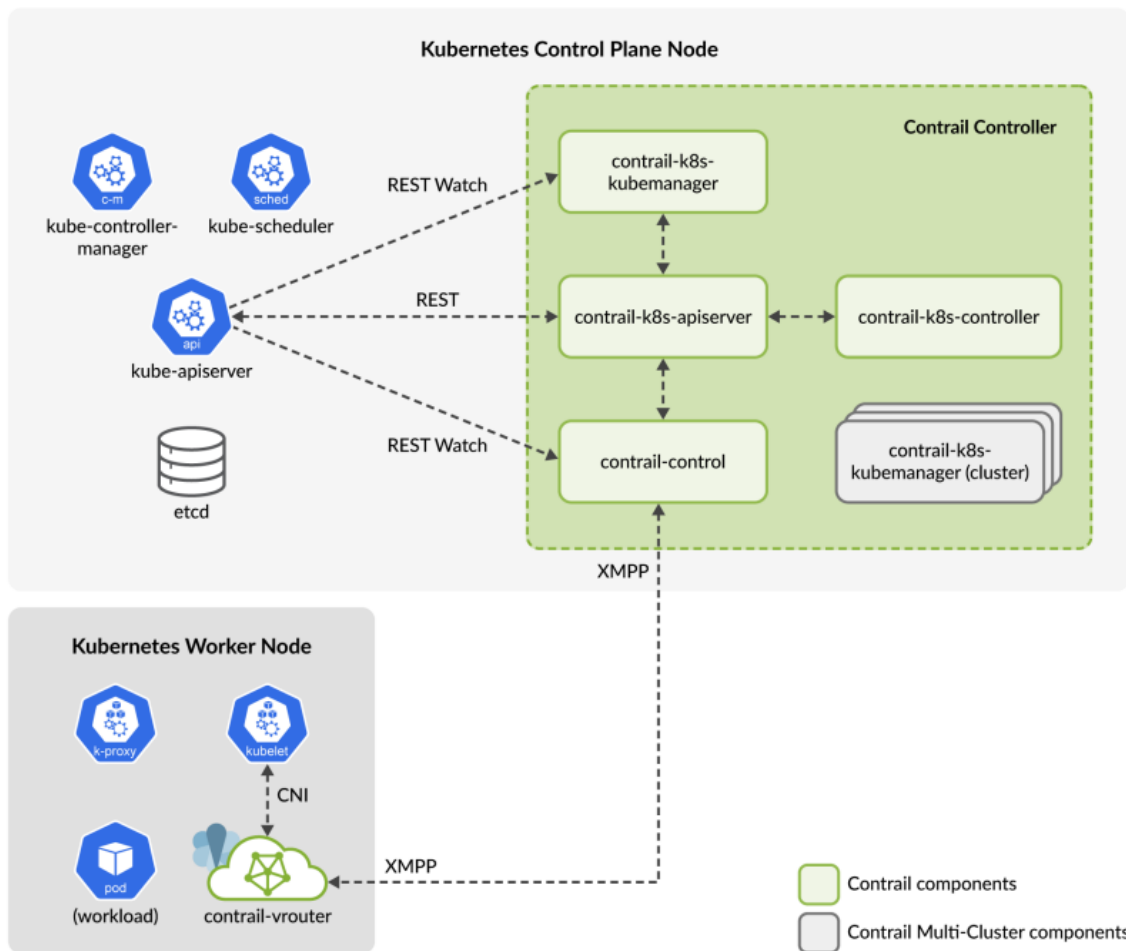
Table 2: CN2 Components (Continued)

Pod Name	Where	Description
1The 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



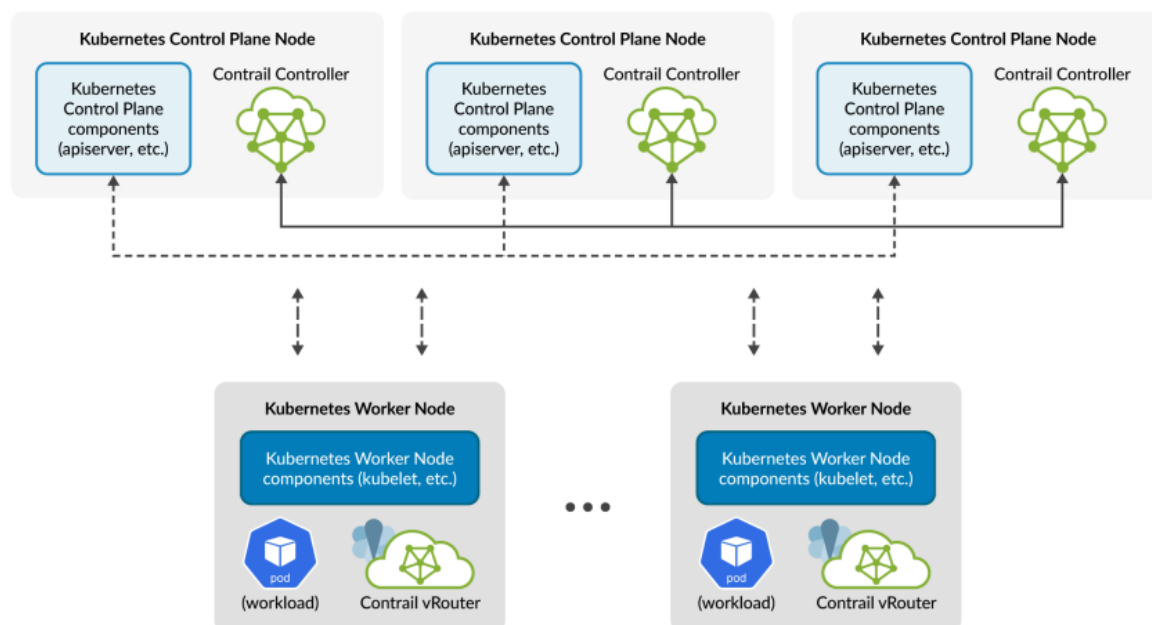
When running on upstream Kubernetes or Rancher RKE2, the Contrail controller stores all CN2 cluster data in the main Kubernetes etcd database by default. When running on Open Shift, the Contrail controller stores all CN2 cluster data in its own Contrail etcd database.

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



«—» REST
 «—» BGP
 «—» REST and XMPP

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.

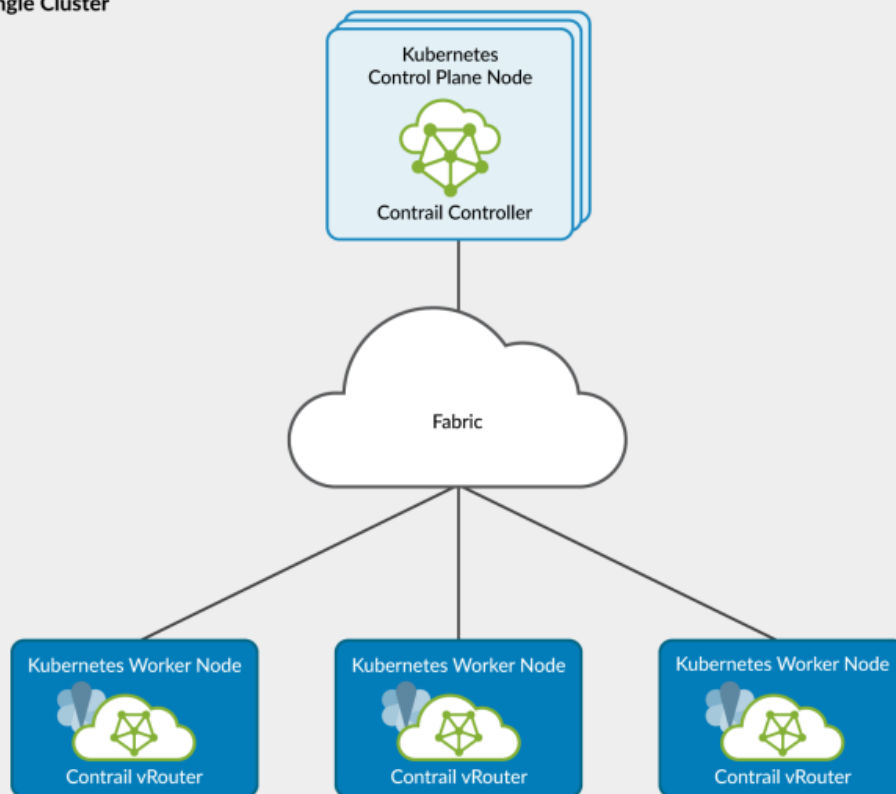
Single Cluster Deployment

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

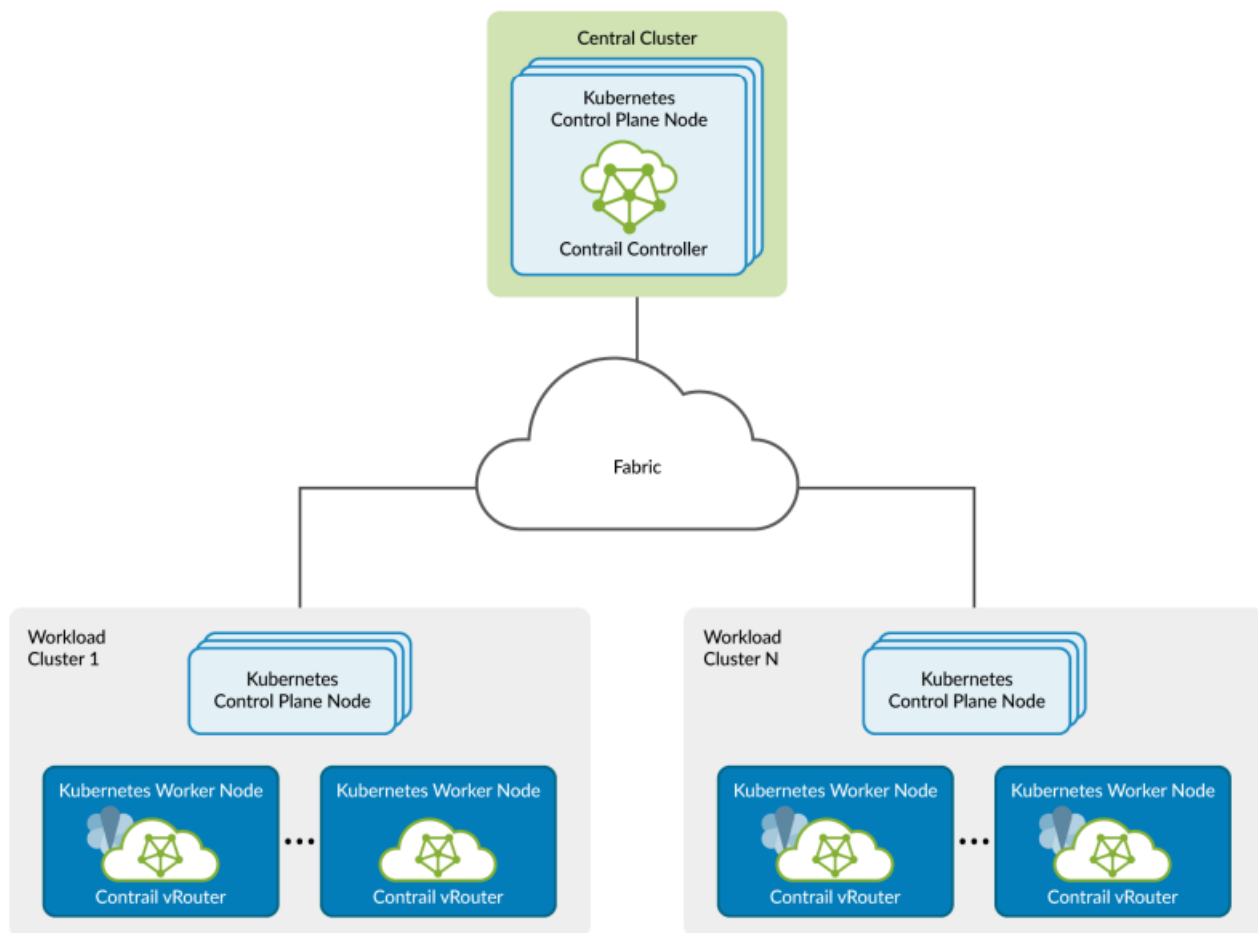
Single Cluster



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.

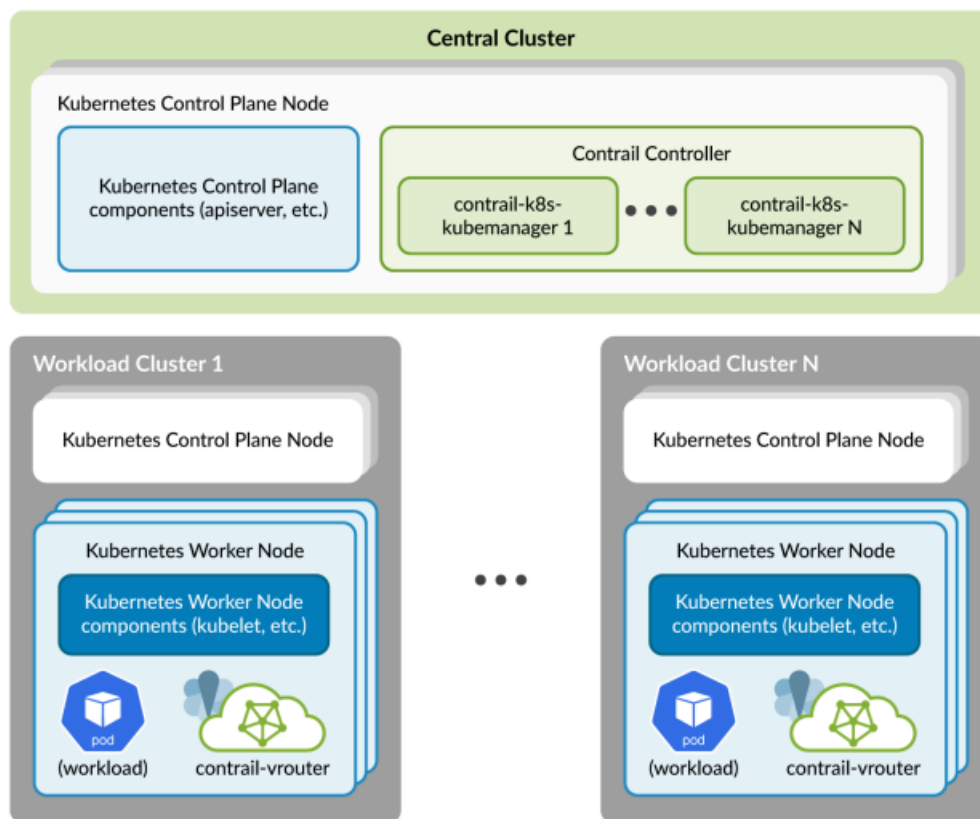
Figure 4: Multi-Cluster Deployment



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



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 singlecluster 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 Upstream Kubernetes Installation with CN2

Machine	CPU	RAM	Storage	Notes
Control Plane Nodes 1	8	32 GB	400 GB	Processor must support the AVX2 instruction set if running DPDK.
Worker Nodes2	4	16 GB	100 GB	Processor must support the AVX2 instruction set if running DPDK.
<p>1. includes nodes in single clusters, central clusters, and distributed workload clusters.</p> <p>2. Based on workload requirements.</p>				

Install

Overview

IN THIS SECTION

- Benefits of Upstream Kubernetes with Contrail | 17

Upstream Kubernetes is an open source version of Kubernetes that is maintained by the Cloud Native Computing Foundation (CNCF). It consists of the core components that provide the infrastructure for container orchestration. It forms the basis for commercial Kubernetes distributions (in other words, it is ‘upstream’ of other distributions).

Upstream Kubernetes does not include any add-on components for monitoring and life-cycle managing your cluster. It’s therefore targeted for organizations that have the ability to put together a usable orchestration solution by themselves. It’s also good for users who want to quickly get a bare bones proof-of-concept installation up and running.

Upstream Kubernetes also does not include a CNI plug-in. After you install a fresh cluster, you’ll need to install a CNI plug-in for that cluster. With CN2, you simply run the supplied Contrail deployer. The Contrail deployer runs in a container and behaves just like any other Kubernetes application. The deployer installs and provides life cycle management for CN2 components.

Once CN2 is installed, you manage it using kubectl and other standard Kubernetes tools. If you also install Contrail Analytics, you’ll get Prometheus, Grafana, and other open source monitoring software installed automatically, with the added benefit that CN2 will work seamlessly with these latter applications with no further configuration necessary.

Benefits of Upstream Kubernetes with Contrail

- Open source Kubernetes platform together with industry-leading CNI
- Install only what you need, fully customizable
- Ideal for roll-your-own and proof-of-concept installations
- Contrail deployer facilitates installation

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. Download the Contrail Networking manifests (“Manifests” on page 38) and extract the tgz onto the host where you plan on running the installation. This host must be able to reach the cluster nodes.
 4. Configure your repository login credentials in the downloaded manifests. Add your repository login credentials to the contrail-manifests-k8s and contrail-tools manifests. See “Configure Repository Credentials” on page 74 for one way to do this.
 5. Configure the 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-testedintegrations.html>).
 - b. Disable transmit checksum offload on any cluster node that is a VM. You must disable the offload in a persistent manner (that survives reboots). There are different ways you can do this, including disabling transmit checksum offload in the VM definition. Use the method that works best in your setup.
 - c. 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/stubresolv.conf`.

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

The cluster nodes in our examples are running Ubuntu.

 - d. 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 77.
 6. Install Contrail tools. See “Install Contrail Tools” on page 36.
 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. 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 3 to 7 for each cluster.

Install Single Cluster Shared Network CN2

SUMMARY

See examples on how to install single cluster CN2 in a deployment where Kubernetes traffic and CN2 traffic share the same network

IN THIS SECTION

- Install Single Cluster Shared Network CN2 Running Kernel Mode Data Plane | 21

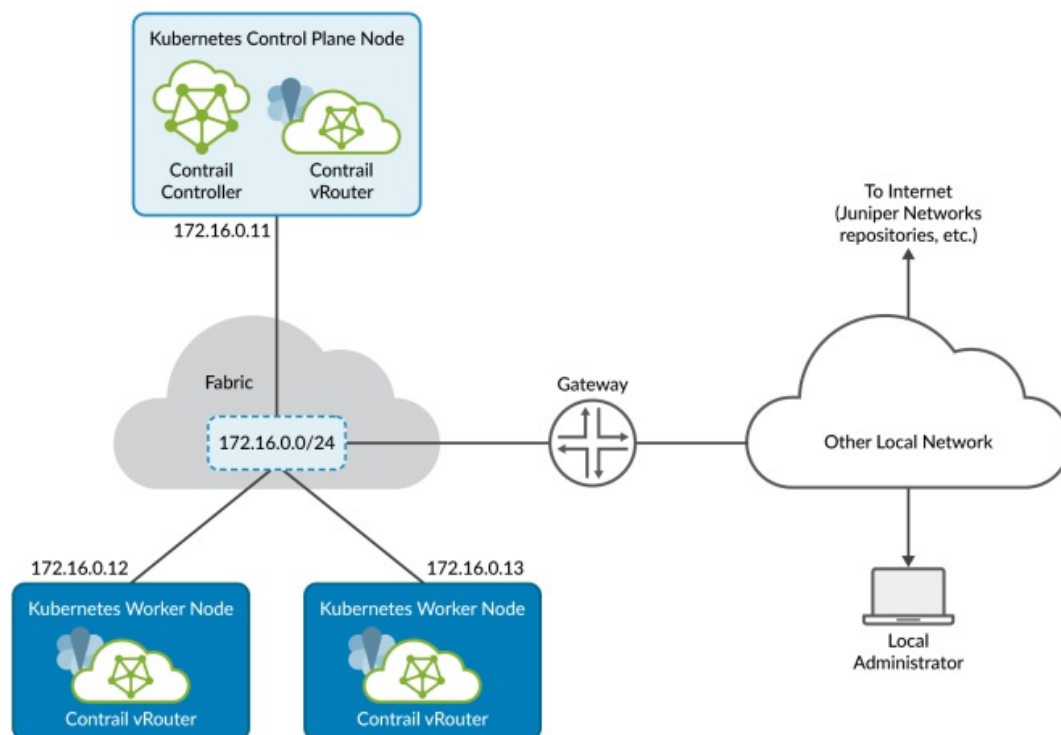
In a single cluster shared network deployment:

- 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.
- Kubernetes and CN2 traffic share a single network.

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

All nodes shown can be VMs or bare metal servers.

Figure 6: Single Cluster Shared Network CN2



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 Shared Network CN2 Running Kernel Mode Data Plane

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

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

1. Create a Kubernetes cluster. You can follow the example procedure in "Create a Kubernetes Cluster" on page 66 or you can use any other method. Create the cluster with the following characteristics:
 - Cluster has no CNI plug-in.
 - Disable Node Local DNS.
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
k8s-cp0	Ready	control-plane,master	65m	v1.20.7
k8s-worker0	Ready	<none>	63m	v1.20.7
k8s-worker1	Ready	<none>	62m	v1.20.7

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	<trimmed>	
contrail-deploy	contrail-k8s-deployer-747689445-7rx52	1/1	Running	0	44m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-control-0	2/2	Running	0	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-k8s-apiserver-6b544788f4-mpk5d	1/1	Running	0	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-k8s-controller-75b8d7b846-rvg7h	1/1	Running	2	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-k8s-kubemanager-6c8b7bd5f5-mwdpj	1/1	Running	5	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-vrouter-masters-pl4zf	3/3	Running	0	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-vrouter-nodes-2tnqq	3/3	Running	0	40m	172.16.0.12	k8s-worker0	<none>	<none>
contrail	contrail-vrouter-nodes-66xnw	3/3	Running	0	40m	172.16.0.13	k8s-worker1	<none>	<none>
kube-system	coredns-657959df74-25sdx	1/1	Running	0	3m19s	10.233.64.2	k8s-cp0	<none>	<none>
kube-system	coredns-657959df74-rprzv	1/1	Running	0	66m	10.233.65.0	k8s-worker0	<none>	<none>
kube-system	dns-autoscaler-b5c786945-pcgsq	1/1	Running	0	66m	10.233.65.1	k8s-worker0	<none>	<none>
kube-system	kube-apiserver-k8s-cp0	1/1	Running	0	69m	172.16.0.11	k8s-cp0	<none>	<none>
kube-system	kube-controller-manager-k8s-cp0	1/1	Running	0	69m	172.16.0.11	k8s-cp0	<none>	<none>
kube-system	kube-proxy-k5mcp	1/1	Running	0	67m	172.16.0.13	k8s-worker1	<none>	<none>
kube-system	kube-proxy-sccjm	1/1	Running	0	67m	172.16.0.11	k8s-cp0	<none>	<none>
kube-system	kube-proxy-wqbt8	1/1	Running	1	67m	172.16.0.12	k8s-worker0	<none>	<none>
kube-system	kube-scheduler-k8s-cp0	1/1	Running	0	69m	172.16.0.11	k8s-cp0	<none>	<none>
kube-system	nginx-proxy-k8s-worker0	1/1	Running	0	67m	172.16.0.12	k8s-worker0	<none>	<none>
kube-system	nginx-proxy-k8s-worker1	1/1	Running	0	67m	172.16.0.13	k8s-worker1	<none>	<none>

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

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 5 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 55.

4. (Optional) Run postflight checks. See “Run Preflight and Postflight Checks” on page 51.

Install Single Cluster Shared Network CN2 Running DPDK Data Plane

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

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

1. Create a Kubernetes cluster. You can follow the example procedure in "Create a Kubernetes Cluster" on page 66 or you can use any other method. Create the cluster with the following characteristics:

- Cluster has no CNI plug-in.
- Disable Node Local DNS.
- 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.

```
kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
k8s-cp0	Ready	control-plane,master	65m	v1.20.7
k8s-worker0	Ready	<none>	63m	v1.20.7
k8s-worker1	Ready	<none>	62m	v1.20.7

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	<trimmed>	
contrail-deploy	contrail-k8s-deployer-747689445-7rx52	1/1	Running	0	44m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-control-0	2/2	Running	0	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-k8s-apiserver-6b544788f4-mpk5d	1/1	Running	0	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-k8s-controller-75b8d7b846-rvg7h	1/1	Running	2	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-k8s-kubemanager-6c8b7bd5f5-mwdpj	1/1	Running	5	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-vrouter-masters-pl4zf	3/3	Running	0	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-vrouter-nodes-2tnqq	3/3	Running	0	40m	172.16.0.12	k8s-worker0	<none>	<none>
contrail	contrail-vrouter-nodes-66xnw	3/3	Running	0	40m	172.16.0.13	k8s-worker1	<none>	<none>
kube-system	coredns-657959df74-25sdx	1/1	Running	0	3m19s	10.233.64.2	k8s-cp0	<none>	<none>
kube-system	coredns-657959df74-rprzv	1/1	Running	0	66m	10.233.65.0	k8s-worker0	<none>	<none>
kube-system	dns-autoscaler-b5c786945-pcgsq	1/1	Running	0	66m	10.233.65.1	k8s-worker0	<none>	<none>
kube-system	kube-apiserver-k8s-cp0	1/1	Running	0	69m	172.16.0.11	k8s-cp0	<none>	<none>
kube-system	kube-controller-manager-k8s-cp0	1/1	Running	0	69m	172.16.0.11	k8s-cp0	<none>	<none>

kube-system	kube-proxy-k5mcp	1/1	Running	0	67m	172.16.0.13	k8s-worker1	<none>	<none>
kube-system	kube-proxy-sccjm	1/1	Running	0	67m	172.16.0.11	k8s-cp0	<none>	<none>
kube-system	kube-proxy-wqbt8	1/1	Running	1	67m	172.16.0.12	k8s-worker0	<none>	<none>
kube-system	kube-scheduler-k8s-cp0	1/1	Running	0	69m	172.16.0.11	k8s-cp0	<none>	<none>
kube-system	nginx-proxy-k8s-worker0	1/1	Running	0	67m	172.16.0.12	k8s-worker0	<none>	<none>
kube-system	nginx-proxy-k8s-worker1	1/1	Running	0	67m	172.16.0.13	k8s-worker1	<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 5 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 55.

5. (Optional) Run portlight checks. See "Run Preflight and Portlight Checks" on page 51.

Install Single Cluster Multi-Network CN2

SUMMARY

See examples on how to install single cluster CN2 in a deployment where Kubernetes traffic and CN2 traffic go over separate networks.

IN THIS SECTION

- Install Single Cluster Multi-Network CN2 Running Kernel Mode Data Plane | 28
- Install Single Cluster Multi-Network CN2 Running DPDK Data Plane | 30

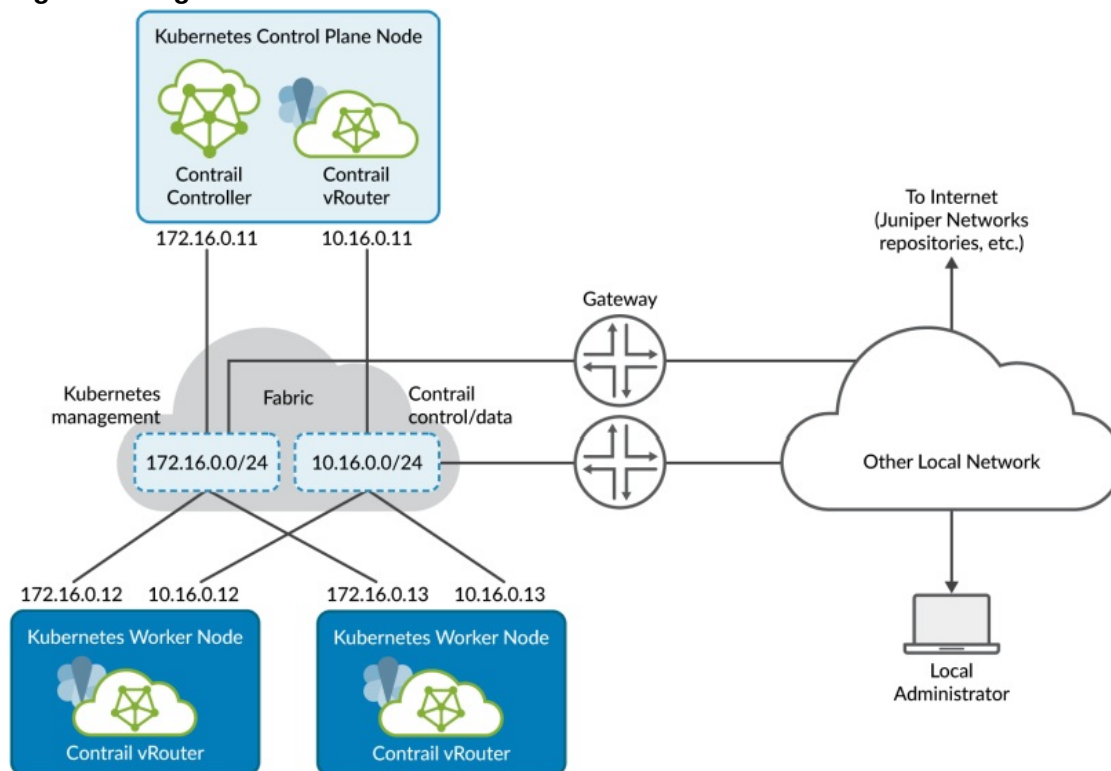
In a single cluster multi-network deployment:

- 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.
- Cluster traffic is separated onto two networks. The Kubernetes control plane traffic traverses one network while Contrail control and data traffic traverse the second network. It's also possible (but less common) to separate traffic onto more than two networks, but this is beyond the scope of these examples.

Figure 7 on page 27 shows the cluster that you'll create if you follow this single cluster multi-network example. The cluster consists of a single control plane node, two worker nodes, and two subnets.

All nodes shown can be VMs or bare metal servers.

Figure 7: Single Cluster Multi-Network CN2



Kubernetes control plane traffic goes over the 172.16.0.0/24 fabric virtual network while Contrail control and data traffic go over the 10.16.0.0/24 fabric virtual network. The fabric networks provide 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 two subnets. 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 Astra-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 Multi-Network CN2 Running Kernel Mode Data Plane

Use this procedure to install CN2 in a single cluster multi-network deployment running a kernel mode data plane. The manifest that you will use in this example procedure is `single-cluster/ single_cluster_deployer_example.yaml`. The procedure assumes that you've placed this manifest into a manifests directory.

1. Create a Kubernetes cluster. You can follow the example procedure in "Create a Kubernetes Cluster" on page 66 or you can use any other method. Create the cluster with the following characteristics:
 - Cluster has no CNI plug-in.
 - Disable Node Local DNS.
2. Modify the `single_cluster_deployer_example.yaml` to configure the Contrail control and data network. You specify the Contrail network using a `contrail-network-config` ConfigMap. The `single_cluster_deployer_example.yaml` manifest contains a commented example on how you can configure a `contrail-network-config` ConfigMap. Either uncomment those lines and specify the appropriate subnet and gateway or copy and paste the following into the manifest.

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: contrail-network-config
  namespace: contrail
data:
  networkConfig: |
    controlDataNetworks:
    - subnet: 10.16.0.0/24
      gateway: 10.16.0.254
```

The subnet and gateway you specify is the Contrail control and data network and gateway, which in our example is the 10.16.0.0/24 network.

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.

```
kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
k8s-cp0	Ready	control-plane,master	65m	v1.20.7
k8s-worker0	Ready	<none>	63m	v1.20.7
k8s-worker1	Ready	<none>	62m	v1.20.7

b. Show the status of the pods.

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

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	<trimmed>	
contrail-deploy	contrail-k8s-deployer-747689445-7rx52	1/1	Running	0	44m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-control-0	2/2	Running	0	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-k8s-apiserver-6b544788f4-mpk5d	1/1	Running	0	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-k8s-controller-75b8d7b846-rvg7h	1/1	Running	2	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-k8s-kubemanager-6c8b7bd5f5-mwdpj	1/1	Running	5	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-vrouter-masters-pl4zf	3/3	Running	0	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-vrouter-nodes-2tnqq	3/3	Running	0	40m	172.16.0.12	k8s-worker0	<none>	<none>
contrail	contrail-vrouter-nodes-66xnw	3/3	Running	0	40m	172.16.0.13	k8s-worker1	<none>	<none>
kube-system	coredns-657959df74-25sdx	1/1	Running	0	3m19s	10.233.64.2	k8s-cp0	<none>	<none>
kube-system	coredns-657959df74-rprzv	1/1	Running	0	66m	10.233.65.0	k8s-worker0	<none>	<none>
kube-system	dns-autoscaler-b5c786945-pcgsq	1/1	Running	0	66m	10.233.65.1	k8s-worker0	<none>	<none>
kube-system	kube-apiserver-k8s-cp0	1/1	Running	0	69m	172.16.0.11	k8s-cp0	<none>	<none>
kube-system	kube-controller-manager-k8s-cp0	1/1	Running	0	69m	172.16.0.11	k8s-cp0	<none>	<none>
kube-system	kube-proxy-k5mcp	1/1	Running	0	67m	172.16.0.13	k8s-worker1	<none>	<none>
kube-system	kube-proxy-sccjm	1/1	Running	0	67m	172.16.0.11	k8s-cp0	<none>	<none>
kube-system	kube-proxy-wqbt8	1/1	Running	1	67m	172.16.0.12	k8s-worker0	<none>	<none>
kube-system	kube-scheduler-k8s-cp0	1/1	Running	0	69m	172.16.0.11	k8s-cp0	<none>	<none>
kube-system	nginx-proxy-k8s-worker0	1/1	Running	0	67m	172.16.0.12	k8s-worker0	<none>	<none>
kube-system	nginx-proxy-k8s-worker1	1/1	Running	0	67m	172.16.0.13	k8s-worker1	<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 5 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 55.
5. (Optional) Run postflight checks. See "Run Preflight and Postflight Checks" on page 51.

I Install Single Cluster Multi-Network CN2 Running DPDK Data Plane

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

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

1. Create a Kubernetes cluster. You can follow the example procedure in "Create a Kubernetes Cluster" on page 66 or you can use any other method. Create the cluster with the following characteristics:
 - Cluster has no CNI plug-in.
 - Disable Node Local DNS.
 - Enable molts version 0.3.1.
2. Modify the `single_cluster_deployer_example.yaml` to configure the Contrail control and data network. You specify the Contrail network using a `contrail-network-config` ConfigMap. The `single_cluster_deployer_example.yaml` manifest contains a commented example on how you can configure a `contrail-network-config` ConfigMap.

Either uncomment those lines and specify the appropriate subnet and gateway or copy and paste the following into the manifest.

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: contrail-network-config
  namespace: contrail
data:
  networkConfig: |
    controlDataNetworks:
    - subnet: 10.16.0.0/24
      gateway: 10.16.0.254
```

The subnet and gateway you specify is the Contrail control and data network and gateway, which in our example is the 10.16.0.0/24 network.

3. 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.

4. 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.

5. Use standard kubectl commands to check on the deployment

a. Show the status of the nodes.

```
kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
k8s-cp0	Ready	control-plane,master	65m	v1.20.7
k8s-worker0	Ready	<none>	63m	v1.20.7
k8s-worker1	Ready	<none>	62m	v1.20.7

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	<trimmed>	
contrail-deploy	contrail-k8s-deployer-747689445-7rx52	1/1	Running	0	44m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-control-0	2/2	Running	0	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-k8s-apiserver-6b544788f4-mpk5d	1/1	Running	0	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-k8s-controller-75b8d7b846-rvg7h	1/1	Running	2	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-k8s-kubemanager-6c8b7bd5f5-mwdpj	1/1	Running	5	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-vrouter-masters-pl4zf	3/3	Running	0	40m	172.16.0.11	k8s-cp0	<none>	<none>
contrail	contrail-vrouter-nodes-2tnqq	3/3	Running	0	40m	172.16.0.12	k8s-worker0	<none>	<none>
contrail	contrail-vrouter-nodes-66xnw	3/3	Running	0	40m	172.16.0.13	k8s-worker1	<none>	<none>
kube-system	coredns-657959df74-25sdx	1/1	Running	0	3m19s	10.233.64.2	k8s-cp0	<none>	<none>
kube-system	coredns-657959df74-rprzv	1/1	Running	0	66m	10.233.65.0	k8s-worker0	<none>	<none>
kube-system	dns-autoscaler-b5c786945-pcgsq	1/1	Running	0	66m	10.233.65.1	k8s-worker0	<none>	<none>
kube-system	kube-apiserver-k8s-cp0	1/1	Running	0	69m	172.16.0.11	k8s-cp0	<none>	<none>
kube-system	kube-controller-manager-k8s-cp0	1/1	Running	0	69m	172.16.0.11	k8s-cp0	<none>	<none>
kube-system	kube-proxy-k5mcp	1/1	Running	0	67m	172.16.0.13	k8s-worker1	<none>	<none>
kube-system	kube-proxy-sccjm	1/1	Running	0	67m	172.16.0.11	k8s-cp0	<none>	<none>
kube-system	kube-proxy-wqbt8	1/1	Running	1	67m	172.16.0.12	k8s-worker0	<none>	<none>
kube-system	kube-scheduler-k8s-cp0	1/1	Running	0	69m	172.16.0.11	k8s-cp0	<none>	<none>
kube-system	nginx-proxy-k8s-worker0	1/1	Running	0	67m	172.16.0.12	k8s-worker0	<none>	<none>
kube-system	nginx-proxy-k8s-worker1	1/1	Running	0	67m	172.16.0.13	k8s-worker1	<none>	<none>

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

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 5 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 55.

6. (Optional) Run postflight checks. See “Run Preflight and Postflight Checks” on page 51.

Install Multi-Cluster Shared Network CN2

SUMMARY

See examples on how to install multi-cluster CN2 in a deployment where Kubernetes traffic and CN2 traffic share the same network within each cluster

IN THIS SECTION

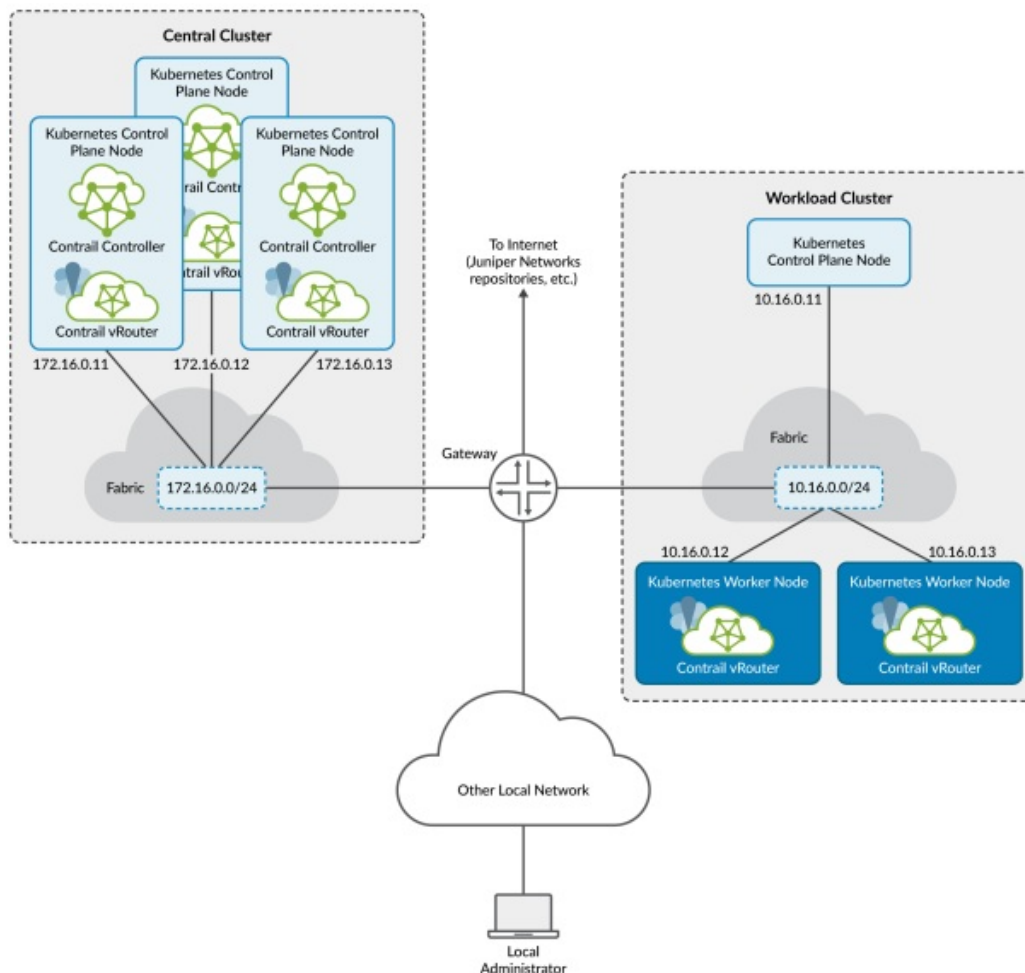
- Install Multi-Cluster Shared Network CN2 35

In a multi-cluster shared network 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.
- Kubernetes and CN2 traffic within each cluster share a single network.

Figure 8 on page 34 shows the cluster you’ll create if you follow the multi-cluster setup. The central cluster consists of 3 Kubernetes control plane nodes that run the Contrail controller. This centralized Contrail controller provides the networking for distributed workload clusters. In this example, there is one distributed cluster that consists of a single control plane node and two worker nodes. The worker nodes on the distributed workload cluster contain the Contrail data plane components.

Figure 8: Multi-Cluster CN2



The central cluster attaches to the 172.16.0.0/24 network while the distributed workload cluster attaches to the 10.16.0.0/24 network. A gateway sitting between the networks provides access to each other and external access for downloading images from Juniper Networks repositories.

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 simplified in the example into a single subnet per cluster. 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.

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 cluster with no CNI plug-in installed and then you'll install CN2 on it.

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.

Install Multi-Cluster Shared Network CN2

Use this procedure to install CN2 in a multi-cluster shared network deployment running a kernel mode data plane.

The manifest that you will use in this example procedure is multi-cluster/ central_cluster_deployer_example.yaml.

The procedure assumes that you've placed this manifest into a manifests directory.

1. Create the central cluster.

Follow the example procedure in “Create a Kubernetes Cluster” on page 66 or you can use any other method.

Create the cluster with the following characteristics:

- Cluster has no CNI plug-in.
- Disable Node Local DNS.

Tailor the procedure with the desired number of control plane and worker nodes accordingly.

2. Install CN2 on the central cluster.

a. Apply the central cluster manifest (central_cluster_deployer_example.yaml). This manifest creates the namespaces and other resources required by the central cluster. It also creates the contrailk8s-deployer deployment, which deploys CN2 and provides life cycle management for the CN2 components.

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

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

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

You've now created the central cluster.

3. Follow “Attach a Workload Cluster” on page 57 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 portlight checks. See “Run Preflight and Portlight Checks” on page 51.

NOTE: Run portlight 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 Contrail Readiness Controller | 37

Contrail tools are implemented within the Contrail Readiness 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 checks” on page 51
- “postflight checks” on page 51
- “CN2 uninstall” on page 55

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 74 for one way to do this.
3. Apply the Contrail Readiness custom resource definitions.

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

4. Create the Config Map from the deployed manifest that you plan to use or have used to install this cluster. Name the Config Map deployed yam.

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

where is the full path to the deployed manifest that you want to apply or have applied.

5. Patch the Config Map with the registry information.

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

6. Create the Contrail Readiness controller.

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

Check that the controller has come up.

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 23.4 | 38
- Contrail Tools in Release 23.4 | 39
- Contrail Analytics in Release 23.4 | 40

Manifests in Release 23.4

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/contrailnetworking/tree/main/releases/23.4/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 Upstream Kubernetes for Release 23.4

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 Upstream Kubernetes for Release 23.4

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

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 23.4

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- <i><release></i> .tar	The kubectl contrailstatus plug-in
contrail-tools/cn2_debug_infra- <i><release></i> .tar	The CN2 debug utility
contrail-tools/uninstall.tar.gz	Deprecated

Contrail Analytics in Release 23.4

The optional Contrail Analytics package is called Analytics Deployed 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.

Monitor

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 contrail status 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 Contrail 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. YYou 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.
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 Node Port 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 82 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 upgrade analytics contrail-analytics-<version>.tgz -f thanos-values.yaml
```

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 3044 3.

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-/P-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-/P-address>:30443`. The default CN2 Web UI username/password is `super/contrail123`.

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 82 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:

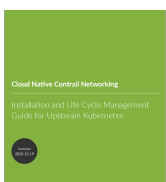
```
helm -n contrail-analytics upgrade analytics contrail-analytics-<version>.tgz
```

or (for upgrading HA)


























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



Documents / Resources

	<p>Juniper NETWORKS Cloud Native Contrail Networking [pdf] Instructions Cloud Native Contrail Networking, Cloud, Native Contrail Networking, Contrail Networking</p>
---	--

References

-  [Kubernetes Documentation | Kubernetes](#)
-  [Juniper Networks – Leader in AI Networking, Cloud, & Connected Security Solutions](#)
-  [Amazon EKS Customers | Managed Kubernetes Service | Amazon Web Services](#)
-  [DPDK](#)
-  [Introduction | RKE2](#)
-  [JFrog](#)
-  [contrail-networking/releases/23.4/k8s at main · Juniper/contrail-networking · GitHub](#)
-  [GitHub - kubernetes-sigs/kubespray: Deploy a Production Ready Kubernetes Cluster](#)
-  [Helm | Installing Helm](#)
-  [Kubernetes](#)
-  [Kubernetes Documentation | Kubernetes](#)
-  [Tasks | Kubernetes](#)
-  [Operating etcd clusters for Kubernetes | Kubernetes](#)
-  [Operating etcd clusters for Kubernetes | Kubernetes](#)
-  [Debugging DNS Resolution | Kubernetes](#)
-  [Debugging DNS Resolution | Kubernetes](#)
-  [storage.googleapis.com/etcd/v3.5.0/etcd-v3.5.0-linux-amd64.tar.gz](#)
-  [Support](#)
-  [Support](#)
-  [Downloads](#)
-  [Downloads](#)
-  [End User License Agreement - Support - Juniper Networks](#)
-  [Secrets - TechLibrary - Juniper Networks](#)
-  [CN2 Tested Integrations | Juniper Networks](#)
-  [Red Hat OpenShift enterprise Kubernetes container platform](#)
- [User Manual](#)