JUNIPER
NETWORKS | Engineering
Simplicity

Cloud Native Contrail Networking

Installation and Life Cycle Management Guide for Upstream Kubernetes

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

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

Table of Contents

3 Monitor

4 Manage

5 Appendix

1

CHAPTER

Introduction

Contrail Overview

SUMMARY

Learn about Cloud Native Contrail Networking.

NOTE: This section is intended to provide a brief and general overview of the Contrail 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.

All references to Kubernetes are made generically and are not intended to single out a particular distribution.

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 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, Cloud Native Contrail Networking 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 Contrail using standard Kubernetes and third-party tools.

- Scale Contrail by scaling the Kubernetes control plane.

- Configure Contrail by using custom resource definitions (CRDs).

- Upgrade Contrail software by applying updated manifests.

- Uninstall Contrail by deleting Contrail namespaces and resources (not supported on OpenShift).

More than a CNI plug-in, Contrail Networking 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 Contrail implementation consists of a set of Contrail controllers residing in the Kubernetes control plane and a CNI plug-in and vRouter on every node. Integrating a full-fledged vRouter alongside the workloads provides Contrail the flexibility to support a wide range of networking requirements, from small single clusters to multi-cluster deployments in large data centers, 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 (via kubevirt)

- Support for DPDK data plane acceleration

Every Kubernetes control plane node contains a Contrail controller that manages a distributed set of vRouter data planes that run on every node. 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.

There is one Contrail controller per Kubernetes control plane node. In a cluster with multiple control plane nodes, the Contrail controllers keep in synchronization by using iBGP to exchange routes. If a control plane node 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, 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 Contrail 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 data center networks.

- Deploy a highly scalable and highly available SDN solution on both upstream and commercial Kubernetes distributions.

- Manage Contrail using familiar, industry-standard tools and practices.

- Leverage the skill set of your existing DevOps engineers to quickly get Contrail up and running.

- Combine with Juniper Networks fabric devices and fabric management solutions or use your own fabric.

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.
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 as a worker node in this document.
Contrail compute node	This is equivalent to a worker node. It is the node where the Contrail vRouter is providing the data plane function.

Table 1: Terminology *(Continued)*

Term	Meaning
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. Contrail supports a centralized network control plane architecture where the routing daemon runs centrally within the Kubernetes control plane 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 Contrail resources. You configure Contrail 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 Contrail that provides the network configuration and network control plane functionality. The Contrail controller resides on the Kubernetes control plane node. This name is purely conceptual – there is no corresponding Contrail controller object or entity in the UI.
Central cluster	In a multi-cluster deployment, this is the central Kubernetes cluster that houses the Contrail controller.
Workload cluster	In a multi-cluster deployment, this is the distributed cluster that contains the workloads.

Components of Contrail

The Contrail 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 Contrail to manage its resources and interact with the rest of the Kubernetes control plane.

- The network control plane represents Contrail'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 6 describes the different Contrail components.

Table 2: Contrail Components

Pod Name		Where	Description
Configuration Plane	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.

Table 2: Contrail Components *(Continued)*

Pod Name		Where	Description
	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.
Control Plane	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.

Table 2: Contrail Components *(Continued)*

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

shows these components in the context of a Kubernetes cluster. For clarity and to reduce clutter, the figure does not show the data plane pods on the Kubernetes control plane node.

Figure 1: Contrail Components
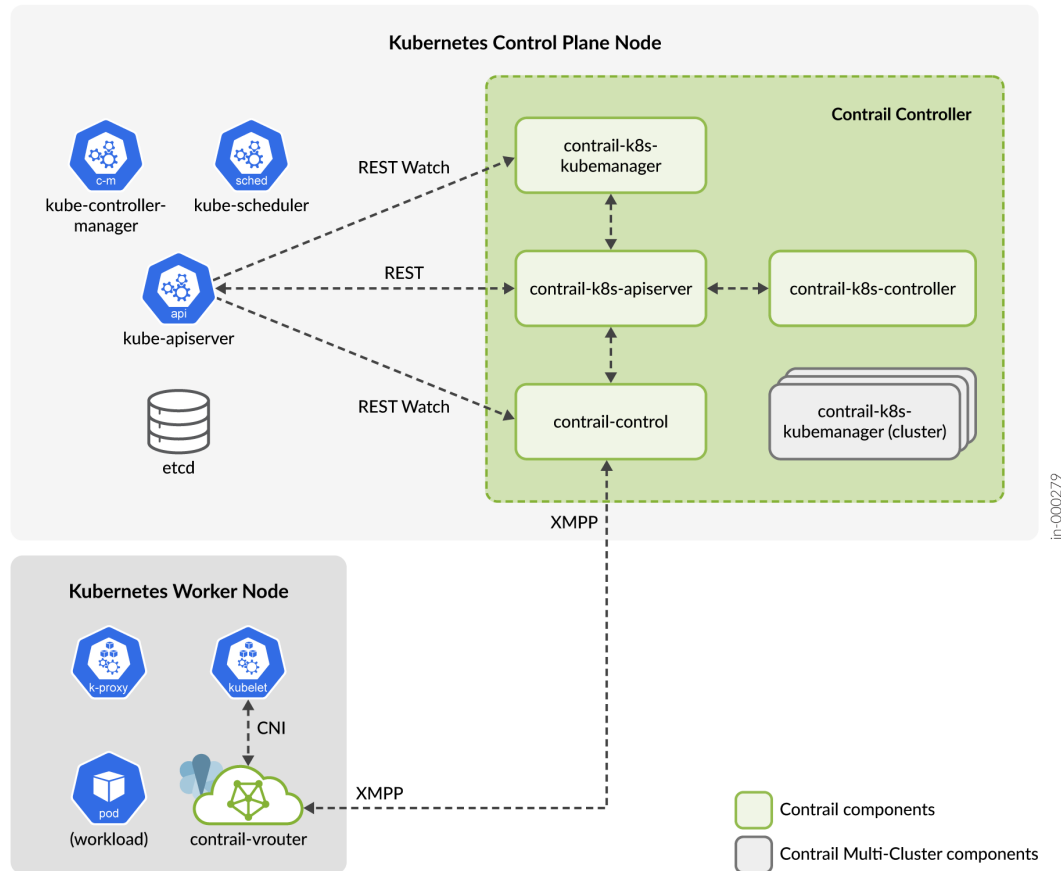
> **NOTE**: Starting in release 22.2, Contrail uses its own etcd database by default when running on OpenShift. In earlier releases and in all Contrail releases running on Upstream Kubernetes, Contrail uses the main Kubernetes etcd database.

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 Contrail 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 how three control plane nodes interact. 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. Every control plane node is identical to every other control plane node in the cluster. Every worker node is identical to every other worker node in the cluster. For redundancy, the vRouter agents on worker nodes always establish XMPP communications with two Contrail controllers.

Figure 2: Multiple Control Plane Nodes
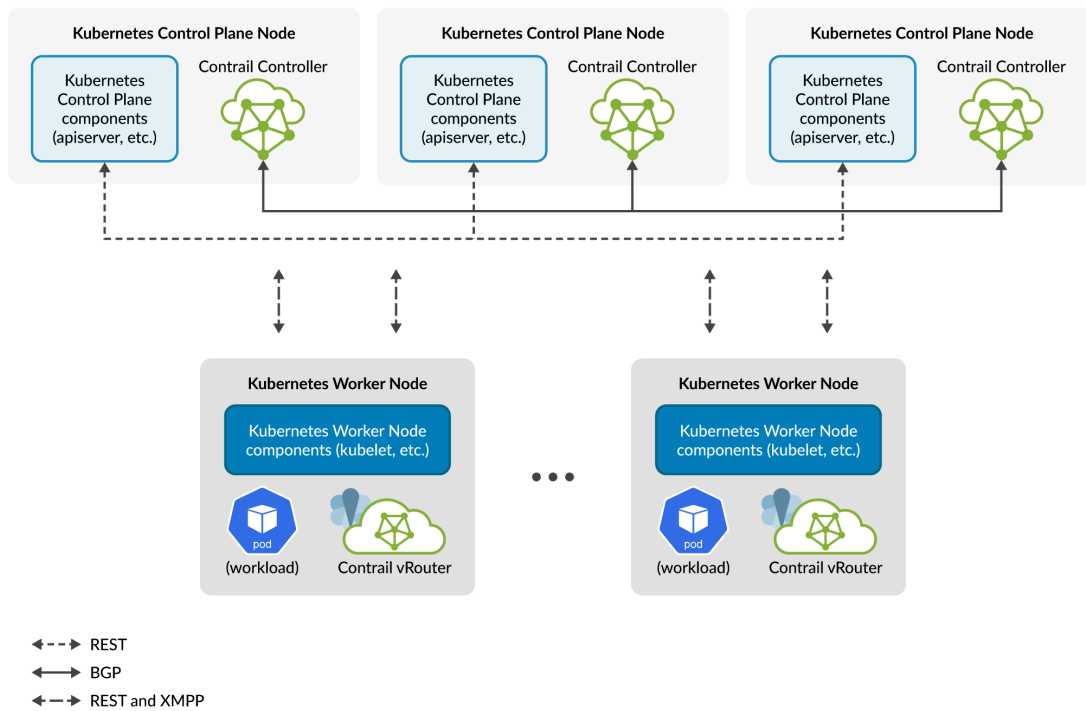
Fabric Requirements

Contrail Networking is agnostic to the fabric technology used. The fabric can consist of Juniper Networks QFX Series switches managed by Juniper Apstra or it can consist of switches from other vendors.

Depending on the type of installation, however, you may need to set up more than one fabric virtual network to connect the cluster nodes. Where necessary, we describe specific fabric requirements in the respective installation sections.

Deployment Models

SUMMARY

IN THIS SECTION

Cloud Native Contrail Networking is available both as an integrated networking platform in a single Kubernetes cluster and as a centralized networking platform to multiple distributed upstream 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 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
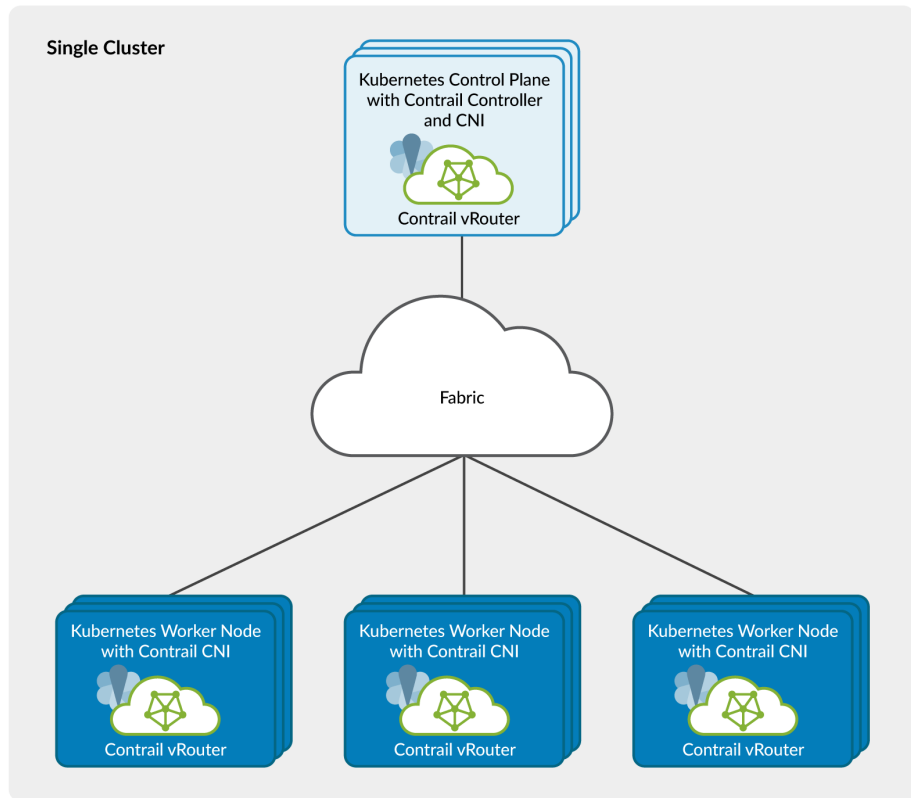
Multi-Cluster Deployment

In a multi-cluster deployment (), 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
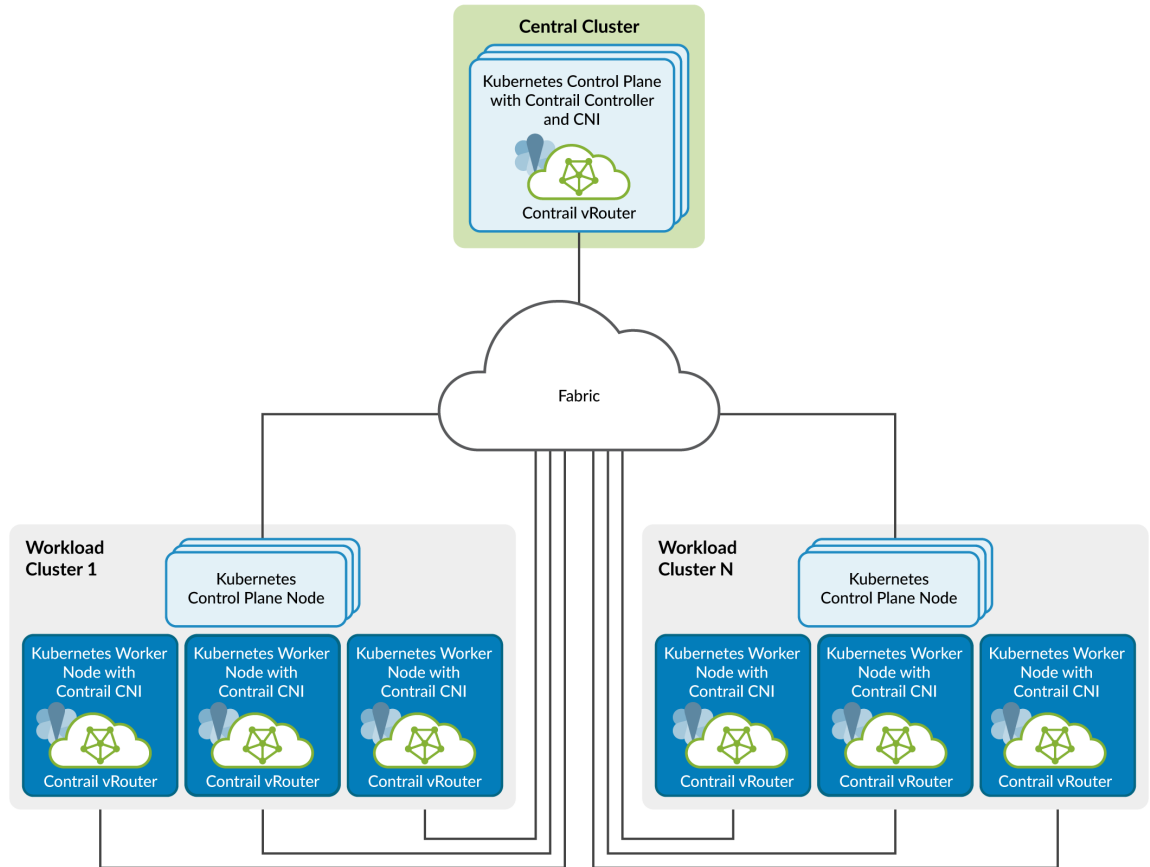
Centralizing the network function in this way makes it not only easier to configure and manage, but also easier to apply consistent network policy and security.

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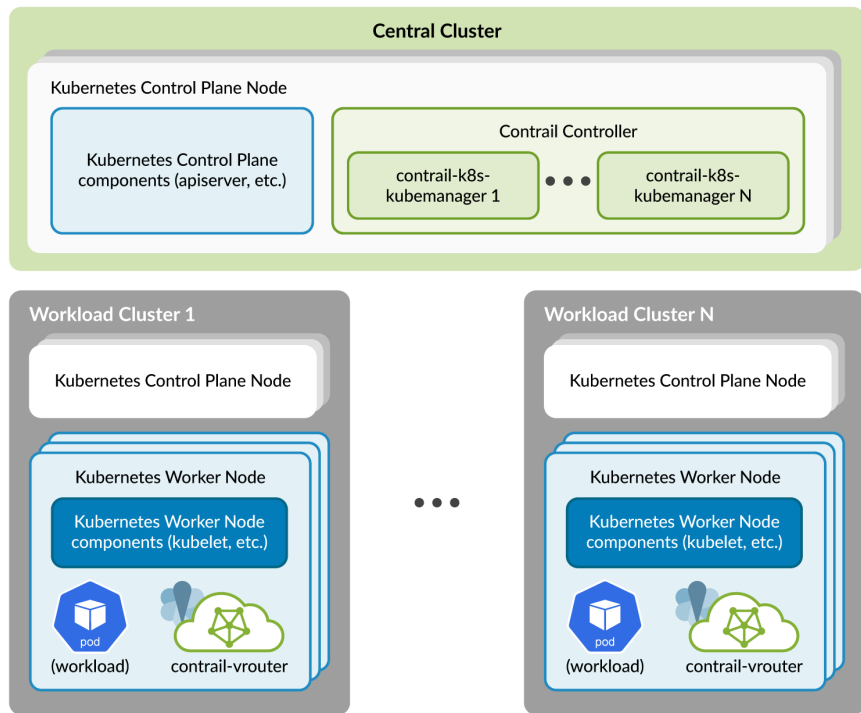
The multi-cluster Contrail controller differs from the single-cluster Contrail controller in two main ways:

- 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 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 Upstream Kubernetes Installation with Contrail

Machine	CPU	RAM	Storage
Control Plane Nodes [1]	8	32 GB	400 GB
Worker Nodes[2]	4	16 GB	100 GB

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

[2] Based on workload requirements.

2

CHAPTER

Install

Overview

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 Contrail, 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 Contrail components.

Once Contrail 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 Contrail 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 Contrail manifests, container packages, and tools.

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 and Tools package ("Manifests and Tools" on page 51) and extract it 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.

   If you're running release 22.1, this step does not apply to you. The release 22.1 manifests are prepopulated with working credentials.

   If you're running release 22.2 or higher, you'll need to add your repository login credentials to the **contrail-manifests-k8s** and **contrail-tools** manifests. See "Configure Repository Credentials" on page 103 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 *Cloud-Native Contrail Networking Supported Platforms* matrix at https://www.juniper.net/documentation/ en_US/release-independent/contrail-cloud-native/topics/reference/cloud-native-contrail-supported-platforms.pdf).

   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

      - 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 106.

6. 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. Change permissions on the **kubectl-contrailstatus** executable and copy it to **/usr/local/bin**.

```
chmod +x tools/kubectl-contrailstatus
```

```
cp tools/kubectl-contrailstatus /usr/local/bin
```

If you're installing a multi-cluster, then repeat steps 3 to 6 for each cluster.

Install Single Cluster Shared Network Contrail

SUMMARY

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

In a single cluster shared network deployment:

- Contrail 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 Contrail traffic share a single network.

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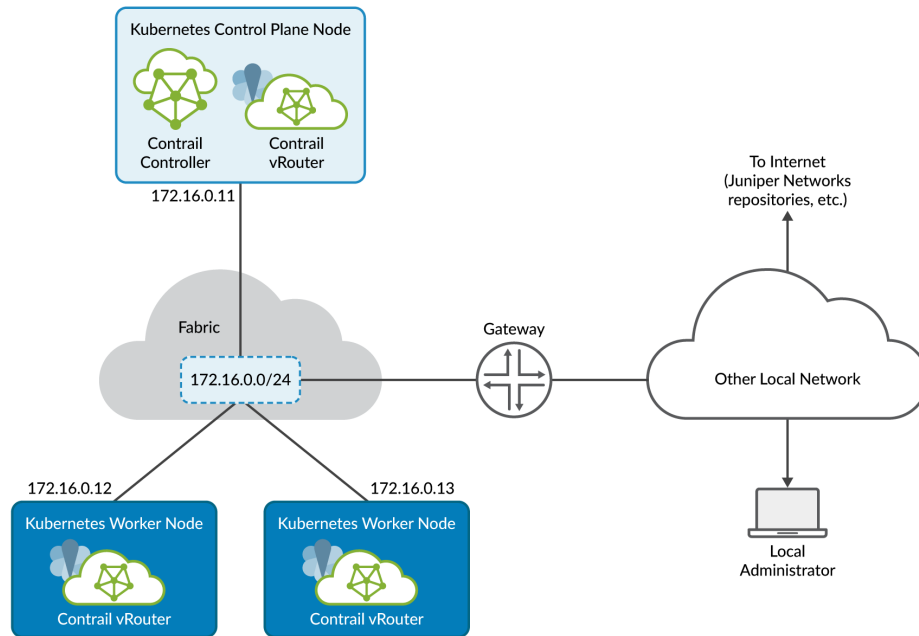
All communication between nodes in the cluster and between nodes and external sites takes place over the single 172.16.0.0/24 fabric virtual network. The fabric network provides the underlay over which the cluster runs.

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 Contrail deployment. You're not limited to the deployment described in this section nor are you limited to using the provided manifests. Contrail 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.

Table 4: Single Cluster Shared Network Examples

Release	Kernel Mode Data Plane	DPDK Data Plane
22.1	"Install Single Cluster Shared Network Contrail Running Kernel Mode Data Plane in Release 22.1" on page 21	"Install Single Cluster Shared Network Contrail Running DPDK Data Plane in Release 22.1" on page 26
22.2	"Install Single Cluster Shared Network Contrail Running Kernel Mode Data Plane in Release 22.2 or Higher" on page 24	"Install Single Cluster Shared Network Contrail Running DPDK Data Plane in Release 22.2" on page 30
22.3 or higher	"Install Single Cluster Shared Network Contrail Running Kernel Mode Data Plane in Release 22.2 or Higher" on page 24	"Install Single Cluster Shared Network Contrail Running DPDK Data Plane in Release 22.3 or Higher" on page 32
NOTE: The provided manifests may not be compatible between releases. Make sure you use the manifests for the release that you're running.		

Install Single Cluster Shared Network Contrail Running Kernel Mode Data Plane in Release 22.1

Use this procedure to install Contrail in a single cluster shared network deployment running a kernel mode data plane in release 22.1.

The manifest that you will use in this example procedure is **single_cluster/deployer.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 91 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 **deployer.yaml** manifest as necessary. The **deployer.yaml** manifest that we provide is a sample that you may need to tailor to your setup.

   Edit **deployer.yaml** to match your setup:

   a. Change the `contrail-vrouter-masters` section as follows.

   The provided **deployer.yaml** file specifies a distinct interface for the Contrail control and data (vRouter) traffic, which is different from our example cluster. In our cluster, the nodes have a single interface that carries both Kubernetes and Contrail control and data traffic. Remove the following lines that describe the distinct vRouter interface.

   ```
   agent:
     virtualHostInterface:
       gateway: 6.6.6.254
   ```

   b. Similarly, remove the following lines in the `contrail-vrouter-nodes` section.

   ```
   agent:
     virtualHostInterface:
       gateway: 6.6.6.254
   ```

   c. Edit the `replicas` configuration to match your setup.

   The provided **deployer.yaml** is intended for a cluster that has three control plane nodes. In our cluster, we have one control plane node. Change the replicas configuration from 3 to 1 for all pods. Here is the result after you make the change.

   ```
   grep replicas deployer.yaml
     replicas: 1
           replicas: 1
           replicas: 1
           replicas: 1
           replicas: 1
   ```

3. Apply the Contrail deployer manifest.

   ```
   kubectl apply -f manifests/deployer.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 hub.juniper.net. For example:

```
user@node:~# ping hub.juniper.net
ping: hub.juniper.net: Temporary failure in name resolution
```

> **NOTE**: Although 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 `/etc/systemd/resolved.conf` to make sure the DNS entry is correct for your network. After you update the configuration, restart the DNS service:

```
user@node:~# systemctl restart systemd-resolved
```

d. If you run into a problem you can't solve or if you made a mistake during the install, simply uninstall Contrail and start over. To uninstall Contrail, see "Uninstall Contrail" on page 71.

Install Single Cluster Shared Network Contrail Running Kernel Mode Data Plane in Release 22.2 or Higher

Use this procedure to install Contrail in a single cluster shared network deployment running a kernel mode data plane in release 22.2 or higher.

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 91 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 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:

```
user@node:~# 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 `/etc/systemd/resolved.conf` to make sure the DNS entry is correct for your network. After you update the configuration, restart the DNS service:

```
user@node:~# systemctl restart systemd-resolved
```

d. If you run into a problem you can't solve or if you made a mistake during the install, simply uninstall Contrail and start over. To uninstall Contrail, see "Uninstall Contrail" on page 71.

4. (Optional) Run postflight checks. See "Run Preflight and Postflight Checks" on page 98.

Install Single Cluster Shared Network Contrail Running DPDK Data Plane in Release 22.1

Use this procedure to install Contrail in a single cluster shared network deployment running a DPDK data plane in release 22.1.

The manifest that you will use in this example procedure is **single_cluster/deployer.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 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. Modify the **deployer.yaml** manifest as necessary. The **deployer.yaml** manifest that we provide is a sample that you may need to tailor to your setup.

   Edit **deployer.yaml** to match your setup:

   a. Change the `contrail-vrouter-dpdk-nodes` section as follows.

      Change the interface name from `eth1` to match the interface name in your setup.

      ```
      virtualHostInterface:
        physicalInterface: eth1
      ```

   b. Change the `contrail-vrouter-masters` section as follows.

      The provided **deployer.yaml** file specifies a distinct interface for the Contrail control and data (vRouter) traffic, which is different from our example cluster. In our cluster, the nodes have a single interface that carries both Kubernetes and Contrail control and data traffic. Remove the following lines that describe the distinct vRouter interface.

      ```
      agent:
        virtualHostInterface:
          gateway: 6.6.6.254
      ```

   c. Similarly, remove the following lines in the `contrail-vrouter-nodes` section.

      ```
      agent:
        virtualHostInterface:
          gateway: 6.6.6.254
      ```

   d. Edit the `replicas` configuration to match your setup.

The provided **deployer.yaml** is intended for a cluster that has three control plane nodes. In our cluster, we have one control plane node. Change the replicas configuration from 3 to 1 for all pods. Here is the result after you make the change.

```
grep replicas deployer.yaml
  replicas: 1
        replicas: 1
        replicas: 1
        replicas: 1
        replicas: 1
```

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, Contrail will use the DPDK configuration specified in the manifest.

4. Apply the Contrail deployer manifest.

   ```
   kubectl apply -f manifests/deployer.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 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 hub.juniper.net. For example:

```
user@node:~# ping hub.juniper.net
ping: hub.juniper.net: Temporary failure in name resolution
```

NOTE: Although 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 `/etc/systemd/resolved.conf` to make sure the DNS entry is correct for your network. After you update the configuration, restart the DNS service:

```
user@node:~# systemctl restart systemd-resolved
```

d. If you run into a problem you can't solve or if you made a mistake during the install, simply uninstall Contrail and start over. To uninstall Contrail, see .

Install Single Cluster Shared Network Contrail Running DPDK Data Plane in Release 22.2

Use this procedure to install Contrail in a single cluster shared network deployment running a DPDK data plane in release 22.2.

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 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> chassis=dpdk-worker
   ```

   By labeling the nodes in this way, Contrail 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:

```
user@node:~# 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 /etc/systemd/resolved.conf to make sure the DNS entry is correct for your network. After you update the configuration, restart the DNS service:

```
user@node:~# systemctl restart systemd-resolved
```

d. If you run into a problem you can't solve or if you made a mistake during the install, simply uninstall Contrail and start over. To uninstall Contrail, see "Uninstall Contrail" on page 71.

5. (Optional) Run postflight checks. See "Run Preflight and Postflight Checks" on page 98.

Install Single Cluster Shared Network Contrail Running DPDK Data Plane in Release 22.3 or Higher

Use this procedure to install Contrail in a single cluster shared network deployment running a DPDK data plane in release 22.3 or higher.

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 91 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, Contrail 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:

```
user@node:~# 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 `/etc/systemd/resolved.conf` to make sure the DNS entry is correct for your network. After you update the configuration, restart the DNS service:

```
user@node:~# systemctl restart systemd-resolved
```

d. If you run into a problem you can't solve or if you made a mistake during the install, simply uninstall Contrail and start over. To uninstall Contrail, see "Uninstall Contrail" on page 71.

5. (Optional) Run postflight checks. See "Run Preflight and Postflight Checks" on page 98.

Install Single Cluster Multi-Network Contrail

SUMMARY

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

In a single cluster multi-network deployment:

- Contrail 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.

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 Contrail
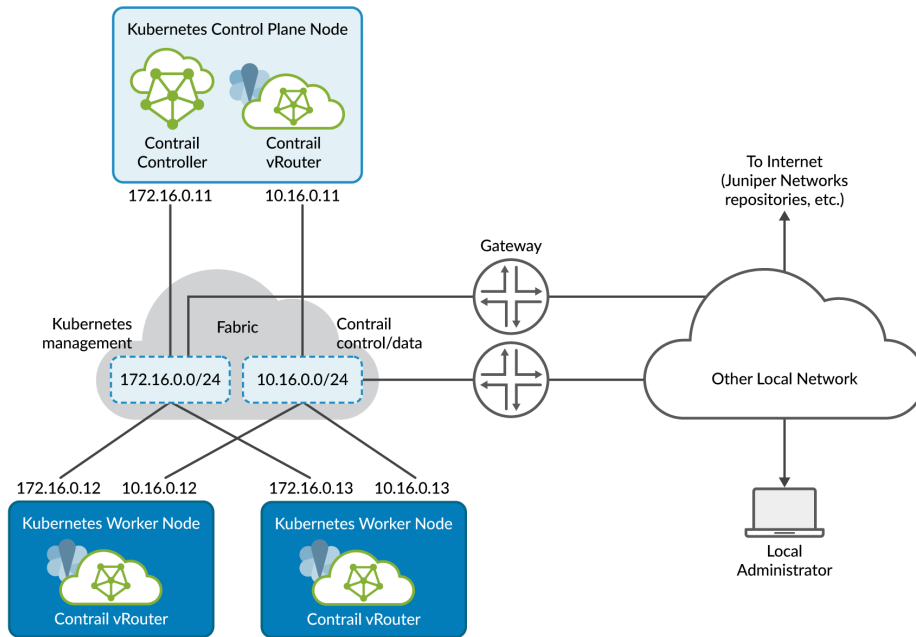
Kubernetes control plane traffic goes over the 172.16.0.0/24 fabric virtual network while Contrail control and data traffic go over the 10.16.0.0/24 fabric virtual network. The fabric networks provide the underlay over which the cluster runs.

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 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 Contrail deployment. You're not limited to the deployment described in this section nor are you limited to using the provided manifests. Contrail 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.

Table 5: Single Cluster Multi-Network Examples

Release	Kernel Mode Data Plane	DPDK Data Plane
22.1	Example not provided	Example not provided
22.2	"Install Single Cluster Multi-Network Contrail Running Kernel Mode Data Plane in Release 22.2 or Higher" on page 37	"Install Single Cluster Multi-Network Contrail Running DPDK Data Plane in Release 22.2" on page 40
22.3 or Higher	"Install Single Cluster Multi-Network Contrail Running Kernel Mode Data Plane in Release 22.2 or Higher" on page 37	"Install Single Cluster Multi-Network Contrail Running DPDK Data Plane in Release 22.3 or Higher" on page 43
NOTE: The provided manifests may not be compatible between releases. Make sure you use the manifests for the release that you're running.		

Install Single Cluster Multi-Network Contrail Running Kernel Mode Data Plane in Release 22.2 or Higher

Use this procedure to install Contrail in a single cluster multi-network deployment running a kernel mode data plane in release 22.2 or higher.

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 91 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
```

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:

```
user@node:~# 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 `/etc/systemd/resolved.conf` to make sure the DNS entry is correct for your network. After you update the configuration, restart the DNS service:

```
user@node:~# systemctl restart systemd-resolved
```

    d. If you run into a problem you can't solve or if you made a mistake during the install, simply uninstall Contrail and start over. To uninstall Contrail, see "Uninstall Contrail" on page 71.

5. (Optional) Run postflight checks. See "Run Preflight and Postflight Checks" on page 98.

Install Single Cluster Multi-Network Contrail Running DPDK Data Plane in Release 22.2

Use this procedure to install Contrail in a single cluster multi-network deployment running a DPDK data plane in release 22.2.

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 91 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. 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> chassis=dpdk-worker
```

By labeling the nodes in this way, Contrail 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 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:

```
user@node:~# 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 `/etc/systemd/resolved.conf` to make sure the DNS entry is correct for your network. After you update the configuration, restart the DNS service:

```
user@node:~# systemctl restart systemd-resolved
```

    d. If you run into a problem you can't solve or if you made a mistake during the install, simply uninstall Contrail and start over. To uninstall Contrail, see "Uninstall Contrail" on page 71.

6. (Optional) Run postflight checks. See "Run Preflight and Postflight Checks" on page 98.

Install Single Cluster Multi-Network Contrail Running DPDK Data Plane in Release 22.3 or Higher

Use this procedure to install Contrail in a single cluster multi-network deployment running a DPDK data plane in release 22.3 or higher.

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 91 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. 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, Contrail 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 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:

```
user@node:~# 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 `/etc/systemd/resolved.conf` to make sure the DNS entry is correct for your network. After you update the configuration, restart the DNS service:

```
user@node:~# systemctl restart systemd-resolved
```

   d. If you run into a problem you can't solve or if you made a mistake during the install, simply uninstall Contrail and start over. To uninstall Contrail, see "Uninstall Contrail" on page 71.

6. (Optional) Run postflight checks. See "Run Preflight and Postflight Checks" on page 98.

Install Multi-Cluster Shared Network Contrail

SUMMARY

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

IN THIS SECTION

In a multi-cluster shared network deployment:

- Contrail 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 Contrail traffic within each cluster share a single network.

Figure 8 on page 47 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 Contrail
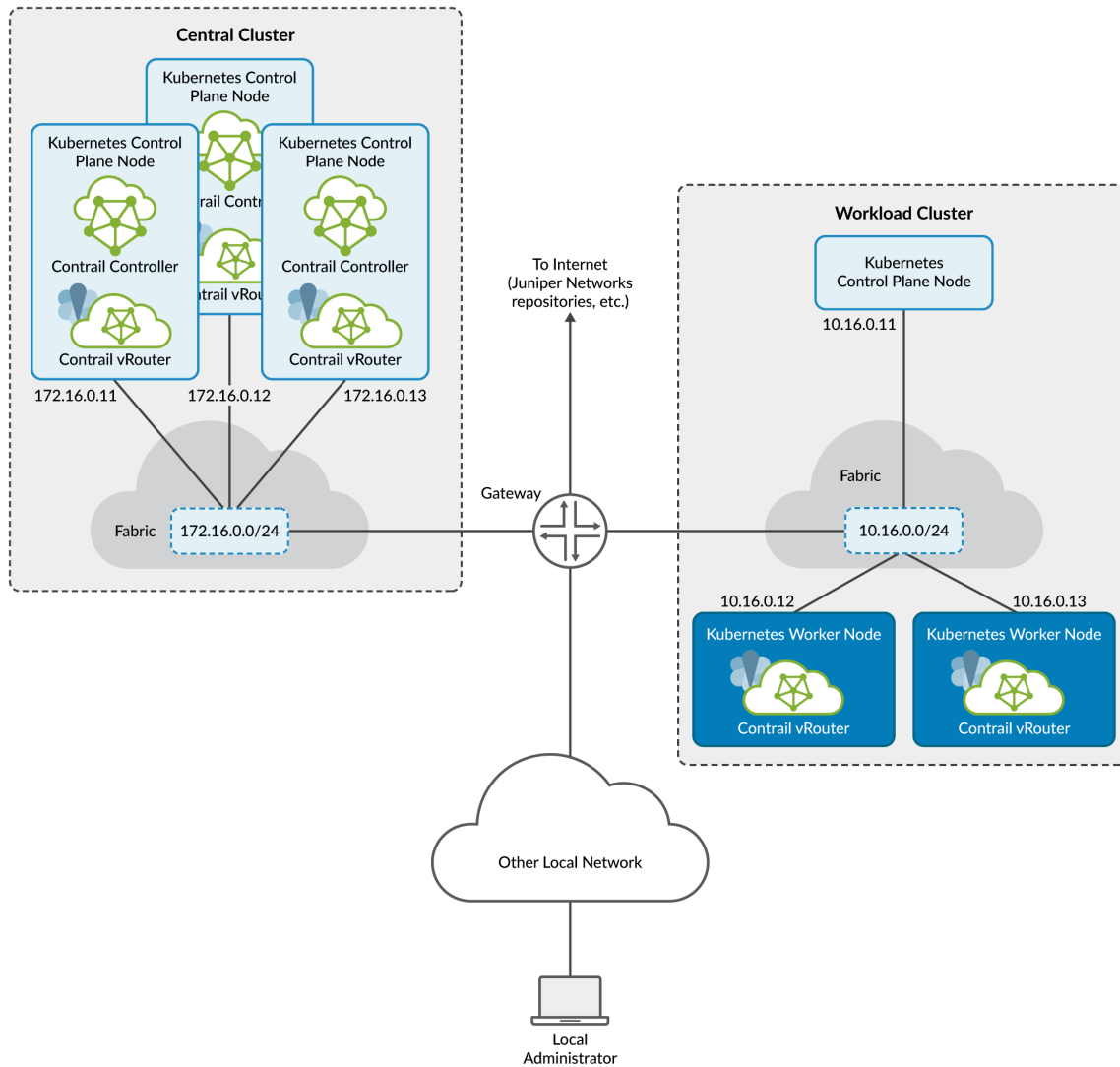
The central cluster attaches to the 172.16.0.0/24 network while the distributed workload cluster attaches to the 10.16.0.0/24 network. A gateway sitting between the networks provides access to each other and external access for downloading images from Juniper Networks repositories.

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 Contrail 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 Contrail on it.

The procedures in this section show basic examples of how you can use the provided manifests to create the specified Contrail deployment. You're not limited to the deployment described in this section nor are you limited to using the provided manifests. Contrail 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.

Table 6: Single Cluster Shared Network Examples

Release	Example
22.1	"Install Multi-Cluster Shared Network Contrail in Release 22.1" on page 48
22.2 or Higher	"Install Multi-Cluster Shared Network Contrail in Release 22.2 or Higher" on page 50

NOTE: The provided manifests may not be compatible between releases. Make sure you use the manifests for the release that you're running.

Install Multi-Cluster Shared Network Contrail in Release 22.1

Use this procedure to install Contrail in a multi-cluster shared network deployment running a kernel mode data plane in release 22.1.

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

1. Create the central cluster.

   Follow the example procedure in 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. Modify the **deployer_ha_central.yaml** manifest as necessary. The **deployer_ha_central.yaml** manifest
   that we provide is a sample that you may need to tailor for your setup.

   The provided **deployer_ha_central.yaml** file specifies distinct networks for data (vRouter) traffic,
   which is different from our example network. In our network, the cluster nodes have a single
   interface that carries all control and data traffic. We'll need to modify **deployer_ha_central.yaml** to
   remove the extra networks.

   Edit **deployer_ha_central.yaml** to remove the following lines that define the `contrail-network-config`
   ConfigMap:

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: contrail-network-config
  namespace: contrail
data:
  networkConfig: |
    controlDataNetworks:
    - subnet: 5.5.5.0/24
      gateway: 5.5.5.254
    - subnet: 5.5.15.0/24
      gateway: 5.5.15.254
```

> **NOTE:** If you're running with 3 control plane nodes in the central cluster as in our cluster
> example, then you do not need to change the `replicas` configuration, which is currently set to
> 3 for all Contrail controller pods. If you're running with a different number of control plane
> nodes, then modify the `replicas` configuration accordingly.

3. Install Contrail on the central cluster.

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

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

4. Follow "Attach a Workload Cluster" on page 74 for your release to create and attach a distributed workload cluster to the central cluster.

5. Repeat step 4 for every workload cluster you want to create and attach.

Install Multi-Cluster Shared Network Contrail in Release 22.2 or Higher

Use this procedure to install Contrail in a multi-cluster shared network deployment running a kernel mode data plane in release 22.2 or higher.

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 91 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 Contrail 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 contrail-

k8s-deployer deployment, which deploys Contrail and provides life cycle management for the Contrail 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 74 for your release 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 98.

NOTE: Run postflight checks from the central cluster only.

Manifests and Tools

SUMMARY

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

IN THIS SECTION

Manifests and Tools in Release 22.1

The Manifests and Tools for Upstream Kubernetes package 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/22.1).

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 package you download is called **contrail-manifests-k8s-22.1.0.93.tgz**.

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

Table 7: Single Cluster Manifests for Upstream Kubernetes for Release 22.1

Manifests	Description
single_cluster/deployer.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 8: Multi-Cluster Manifests for Upstream Kubernetes for Release 22.1

Manifests	Description
multi-cluster/central/deployer_ha_central.yaml	Contrail deployer and necessary resources for the central cluster in a multi-cluster setup.
multi-cluster/dist-dpdk/ distributed_cluster_deployer.yaml	Contrail deployer and necessary resources to support DPDK in distributed workload clusters.
multi-cluster/dist-dpdk/deployer_ha_dpdk.yaml	Contrail DPDK vRouter for the distributed workload clusters.

Table 8: Multi-Cluster Manifests for Upstream Kubernetes for Release 22.1 *(Continued)*

Manifests	Description
multi-cluster/dist-kernel/ distributed_cluster_deployer.yaml	Contrail deployer and necessary resources to support kernel-mode in distributed workload clusters.
multi-cluster/dist-kernel/deployer_ha_kernel.yaml	Contrail kernel-mode vRouter for the distributed workload clusters.

The following table lists miscellaneous tools in that package. These tools are only available from the Juniper Networks software download site.

Table 9: Tools for Upstream Kubernetes for Release 22.1

Tools	Description
contrail-tools/preflight.sh	Performs preflight checks on the node running the script
contrail-tools/uninstall.tgz	Uninstalls Contrail
contrail-tools/kubectl-contrailstatus	The kubectl contrailstatus plug-in

Contrail Analytics

The optional Contrail Analytics package 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.

The release 22.1 package is called **contrail-analytics-22.1.0.93.tgz**. See "Install Contrail Analytics in Release 22.1" on page 60.

Manifests and Tools in Release 22.2

The Manifests and Tools for Upstream Kubernetes package 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/22.2).

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 package you download is called **contrail-manifests-k8s-22.2.0.93.tgz**.

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

Table 10: Single Cluster Manifests for Upstream Kubernetes for Release 22.2

Manifests	Description
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 11: Multi-Cluster Manifests for Upstream Kubernetes for Release 22.2

Manifests	Description
multi-cluster/central_cluster_deployer_example.yaml	Contrail deployer and necessary resources for the central cluster in a multi-cluster setup.
multi-cluster/distributed_cluster_deployer_example.yaml	Contrail deployer and necessary resources for distributed workload clusters in a multi-cluster setup.
multi-cluster/distributed_cluster_vrouter_example.yaml	Contrail vRouter for the distributed workload clusters in a multi-cluster setup.

The following table lists miscellaneous tools in that package. These tools are only available from the Juniper Networks software download site.

Table 12: Tools for Upstream Kubernetes for Release 22.2

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/crds	ContrailReadiness custom resource definitions for preflight and postflight checks
contrail-tools/uninstall.tgz	Uninstalls Contrail
contrail-tools/kubectl-contrailstatus	The kubectl contrailstatus plug-in

Contrail Analytics

The optional Contrail Analytics package 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.

The release 22.2 package is called **contrail-analytics-22.2.0.93.tgz**. See "Install Contrail Analytics in Release 22.2 or Higher" on page 61.

Manifests and Tools in Release 22.3

IN THIS SECTION

The Manifests and Tools for Upstream Kubernetes package 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/22.3).

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 package you download is called **contrail-manifests-k8s-22.3.0.71.tgz**.

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

Table 13: Single Cluster Manifests for Upstream Kubernetes for Release 22.3

Manifests	Description
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 14: Multi-Cluster Manifests for Upstream Kubernetes for Release 22.3

Manifests	Description
multi-cluster/central_cluster_deployer_example.yaml	Contrail deployer and necessary resources for the central cluster in a multi-cluster setup.
multi-cluster/distributed_cluster_deployer_example.yaml	Contrail deployer and necessary resources for distributed workload clusters in a multi-cluster setup.
multi-cluster/distributed_cluster_vrouter_example.yaml	Contrail vRouter for the distributed workload clusters in a multi-cluster setup.

The following table lists miscellaneous tools in that package. These tools are only available from the Juniper Networks software download site.

Table 15: Tools for Upstream Kubernetes for Release 22.3

Tools	Description
contrail-tools/contrail-readiness/contrail-readiness-controller.yaml	The ContrailReadiness controller that runs preflight and postflight checks

Table 15: Tools for Upstream Kubernetes for Release 22.3 *(Continued)*

Tools	Description
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/crds	ContrailReadiness custom resource definitions for preflight and postflight checks
contrail-tools/uninstall.tgz	Uninstalls Contrail
contrail-tools/kubectl-contrailstatus	The kubectl contrailstatus plug-in

Contrail Analytics

The optional Contrail Analytics package 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.

The release 22.3 package is called **contrail-analytics-22.3.0.71.tgz**. See .

3
CHAPTER

Monitor

Overview

You can monitor Contrail the same way you monitor other Kubernetes components, using kubectl, Lens, or any other standard Kubernetes tool.

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.

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

Install Contrail Analytics

SUMMARY

Learn how to install Contrail Analytics.

IN THIS SECTION

Contrail Analytics packages popular open source software such as Prometheus, Grafana, and Fluentd together with Contrail 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.

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

> **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.

Install Contrail Analytics in Release 22.1

Use this procedure to install Contail Analytics in release 22.1.

1. Locate the Contrail Analytics package that you downloaded.

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

2. Create a namespace for the Contrail Analytics pods.

```
kubectl create namespace contrail-analytics
```

3. Create a secret based on your repository login credentials. We'll call this secret **juniper-hub-auth** in this example.

```
kubectl -n contrail-analytics create secret docker-registry juniper-hub-auth --docker-server=hub.juniper.net --
docker-username=<userName>  --docker-password=<password>
```

4. Populate the **image-pull-secret-values.yaml** file with references to the secret you just created.

   a. Extract the **image-pull-secret-values.yaml** file from the Contrail Analytics package.

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

   b. Replace all occurrences of _SECRET_NAME_ in that file with the name of the secret you created.

```
sed -i 's/_SECRET_NAME_/juniper-hub-auth/' image-pull-secret-values.yaml
```

5. Install the Contrail Analytics package, referencing the **image-pull-secret-values.yaml** file you just updated.

```
helm -n contrail-analytics install analytics contrail-analytics-<version>.tgz -f image-pull-secret-values.yaml --set
externalIP=<ipAddress>
```

where *<ipAddress>* is the optional externally-routable virtual IP address that you want to use to access the analytics components. You would use this IP address, for example, to access Grafana from your web browser. In the example networks shown in this document, this IP address would be an unused address in the 172.16.0.0/24 (machine) subnet. If you want to access this address from

outside this subnet, then you may need to create a NAT rule to allow external browsers to reach this address.

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

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

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

7. After you install Contrail Analytics, you can access Grafana by pointing your browser to https:// *<external-IP-address>*/grafana/. Be sure to include the trailing /.

8. To uninstall Contrail Analytics:

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

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

Install Contrail Analytics in Release 22.2 or Higher

Use this procedure to install Contail Analytics in release 22.2 or higher.

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 considered a Tech Preview feature.

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 --set externalIP=<ipAddress>
```

where *<ipAddress>* is the optional externally-routable virtual IP address that you want to use to access the analytics components. You would use this IP address, for example, to access Grafana from your web browser. In the example networks shown in this document, this IP address would be an unused address in the 172.16.0.0/24 (machine) subnet. If you want to access this address from

outside this subnet, then you may need to create a NAT rule to allow external browsers to reach this address.

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 111 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 --set externalIP=<ipAddress>
```

where *<ipAddress>* is the optional externally-routable virtual IP address that you want to use to access the analytics components. You would use this IP address, for example, to access Grafana from your web browser. In the example networks shown in this document, this IP address would be an unused address in the 172.16.0.0/24 (machine) subnet. If you want to access this address from outside this subnet, then you may need to create a NAT rule to allow external browsers to reach this address.

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 by pointing your browser to https://*<external-IP-address>*/grafana/. Be sure to include the trailing /.

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 --set externalIP=<ipAddress>
```

or (for upgrading HA)

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

Kubectl Contrailstatus

IN THIS SECTION

Syntax

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

Description

This command displays the status' of various Contrail 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 Contrail.

kubectl contrailstatus resource virtualnetwork	Displays the status' of the various VirtualNetworks in Contrail.
kubectl contrailstatus --all	Displays the status' of the Configuration/Control/Data planes and the BGP and XMPP relationships.
kubectl contrailstatus version	Displays the contrailstatus version.

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

lists the output fields for the `kubectl contrailstatus` command.

Table 16: 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.

Table 16: kubectl contrailstatus Output Fields *(Continued)*

Field Name	Field Description
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

This command is supported in the initial release.

4

CHAPTER

Manage

Manage Single Cluster Contrail

SUMMARY

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

Overview

The way that you manage a Kubernetes cluster does not change when Contrail is the CNI plug-in. Once Contrail is installed, Contrail 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. Contrail works seamlessly with kubectl and other tools such as Prometheus and Grafana.

See the Appendix for examples of how to perform some common tasks on a Kubernetes cluster. These examples are not specific to Contrail and are provided with no warranty.

The remainder of this chapter contains tasks that are specific to Contrail.

Upgrade Contrail

Use this procedure to upgrade Contrail.

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

1. Download the manifests for the new release.
2. Locate the (old) manifest(s) that you used to create the existing Contrail 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 Contrail.

   ```
   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 Contrail

Use this procedure to uninstall Contrail.

We supply a script that uninstalls Contrail resources associated with Contrail namespaces. The script deletes 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

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

1. 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 step 6 in "Install Contrail Analytics in Release 22.2 or Higher" on page 61.

2. Delete any other resources and namespaces (for example, overlay networks) that you created after you installed Contrail.

3. Locate and extract the uninstall script from the downloaded Contrail Manifests and Tools package.

4. Run the script to uninstall Contrail. You must run the script from the extracted **uninstall** directory.

   ```
   ./uninstall.sh
   ```

   The uninstall should complete after a few minutes.

   If the script doesn't complete, type **<CTRL> c** to stop the uninstall.

   It's likely that one or more namespaces are stuck in the Terminating state. To check the namespaces:

   ```
   kubectl get ns
   ```

If you see a namespace with a status of Terminating, get more information on that namespace:

```
kubectl get ns <namespace> -o yaml
```

Look in the **conditions** section of the output to see what's blocking the uninstall. For example:

```
kubectl get ns contrail -o yaml
<trimmed>
status:
  conditions:
  - lastTransitionTime: "2022-04-25T22:09:51Z"
    message: 'Discovery failed for some groups, 2 failing: unable to retrieve the
      complete list of server APIs: core.contrail.juniper.net/v1alpha1: the server
      is currently unable to handle the request, idallocator.contrail.juniper.net/v1alpha1:
      the server is currently unable to handle the request'
    reason: DiscoveryFailed
<trimmed>
```

In this example output, the problem is the API services. Find out which API services are not available.

```
kubectl get apiservices
NAME                                     SERVICE                      AVAILABLE                AGE
<trimmed>
v1alpha1.core.contrail.juniper.net       contrail-system/contrail-api  False (MissingEndpoints)  57m
v1alpha1.idallocator.contrail.juniper.net  contrail-system/contrail-api  False (MissingEndpoints)  57m
<trimmed>
```

Delete those API services with False availability.

```
kubectl delete apiservice v1alpha1.core.contrail.juniper.net
apiservice.apiregistration.k8s.io "v1alpha1.core.contrail.juniper.net" deleted
```

```
kubectl delete apiservice v1alpha1.idallocator.contrail.juniper.net
apiservice.apiregistration.k8s.io "v1alpha1.idallocator.contrail.juniper.net" deleted
```

Rerun the uninstall script. You'll likely see errors because the script will try to delete resources that may already have been deleted when you ran the script earlier. For example:

```
./uninstall.sh
namespace "contrail" deleted
Error from server (NotFound): error when deleting "contrail.yaml": serviceaccounts "contrail-serviceaccount" not
found
Error from server (NotFound): error when deleting "contrail.yaml": clusterroles.rbac.authorization.k8s.io "contrail-
role" not found
Error from server (NotFound): error when deleting "contrail.yaml": clusterrolebindings.rbac.authorization.k8s.io
"contrail-rolebinding" not found
error: the server doesn't have a resource type "gvc"
error: the server doesn't have a resource type "gsc"
namespace "contrail-system" deleted
clusterrole.rbac.authorization.k8s.io "contrail-system-role" deleted
clusterrolebinding.rbac.authorization.k8s.io "contrail-system-rolebinding" deleted
namespace "contrail-deploy" deleted
clusterrole.rbac.authorization.k8s.io "contrail-deploy-role" deleted
clusterrolebinding.rbac.authorization.k8s.io "contrail-deploy-rolebinding" deleted
customresourcedefinition.apiextensions.k8s.io "apiservers.configplane.juniper.net" deleted
customresourcedefinition.apiextensions.k8s.io "controls.controlplane.juniper.net" deleted
customresourcedefinition.apiextensions.k8s.io "controllers.configplane.juniper.net" deleted
customresourcedefinition.apiextensions.k8s.io "kubemanagers.configplane.juniper.net" deleted
customresourcedefinition.apiextensions.k8s.io "vrouters.dataplane.juniper.net" deleted
customresourcedefinition.apiextensions.k8s.io "network-attachment-definitions.k8s.cni.cncf.io" deleted
```

Manage Multi-Cluster Contrail

SUMMARY

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

IN THIS SECTION

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 Contrail" on page 69.

Attach a Workload Cluster

IN THIS SECTION

Attach a Workload Cluster Running Kernel Mode Data Plane in Release 22.1

Use this procedure to create and attach a distributed workload cluster (running a kernel mode data plane) to the central cluster in release 22.1. 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 **multi-cluster/dist-kernel/distributed_cluster_deployer.yaml** and **multi-cluster/dist-kernel/deployer_ha_kernel.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 Kubernetes Cluster" on page 91 or use your own methods to create a cluster. The cluster should have the following characteristics:

   - Cluster has no CNI plug-in.

   - Disable Node Local DNS.

   - 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 Contrail components on the distributed workload cluster.

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

```
user@workload:~/contrail$ kubectl create ns contrail-deploy
```

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

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

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**.

```
user@workload:~/contrail$ 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 Contrail components.

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

```
user@workload:~/contrail$ kubectl apply -f manifests/distributed_cluster_deployer.yaml
```

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

```
user@workload:~/contrail# kubectl get pods -n contrail-deploy
NAME                                         READY   STATUS    RESTARTS   AGE
contrail-k8s-deployer-6458859585-xhwx6       1/1     Running   0          6m
```

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.

```
user@central:~/contrail$ 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**).

```
user@central:~/contrail$ 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
    nodeSelector:
      node-role.kubernetes.io/master: ""
  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>
```

explains the parameters that you need to set.

Table 17: Kubemanager CRD

Parameter	Meaning	Example
name	The name of the custom resource.	kubemanager-cluster1
image	The repository where you pull images	hub.juniper.net/cn2/contrail-k8s-kubemanager:22.1.0.93
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 17: 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.

```
user@central:~/contrail$ 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. On the workload cluster, modify the **deployer_ha_kernel.yaml** manifest as necessary. The **deployer_ha_kernel.yaml** manifest that we provide is a sample that you may need to tailor to your setup.

The provided **deployer_ha_kernel.yaml** file specifies different networks for the different cluster nodes, which is different from our example cluster. In our cluster, the cluster nodes all connect to the same data network. We'll need to modify **deployer_ha_kernel.yaml** to remove the extra networks and extra node configuration.

a. Remove the extra networks.

Edit **deployer_ha_kernel.yaml** to remove the following lines that define the `contrail-network-config` ConfigMap:

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: contrail-network-config
  namespace: contrail
data:
  networkConfig: |
    controlDataNetworks:
    - subnet: 5.5.6.0/24
      gateway: 5.5.6.254
    - subnet: 5.5.7.0/24
      gateway: 5.5.7.254
    - subnet: 5.5.8.0/24
      gateway: 5.5.8.254
    - subnet: 5.5.14.0/24
      gateway: 5.5.14.254
```

b. Remove the following lines that define the `contrail-vrouter-nodes-7` vRouter:

```
---
apiVersion: dataplane.juniper.net/v1alpha1
kind: Vrouter
metadata:
  name: contrail-vrouter-nodes-7
  namespace: contrail
spec:
  clusterName: distributed-cluster
  common:
    affinity:
      nodeAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
```

```
            nodeSelectorTerms:
            - matchExpressions:
              - key: chassis
                operator: In
                values:
                - "5d10s3-node3"
      imagePullSecrets:
        - name: registrypullsecret
      containers:
      - image: hub.juniper.net/cn2/contrail-vrouter-agent:22.1.0.93
        name: contrail-vrouter-agent
      - image: hub.juniper.net/cn2/contrail-init:22.1.0.93
        name: contrail-watcher
      - image: hub.juniper.net/cn2/contrail-telemetry-exporter:22.1.0.93
        name: contrail-vrouter-telemetry-exporter
      initContainers:
      - image: hub.juniper.net/cn2/contrail-init:22.1.0.93
        name: contrail-init
      - image: hub.juniper.net/cn2/contrail-cni-init:22.1.0.93
        name: contrail-cni-init
    maxUnavailablePercentage: 100%
```

c. Similarly, remove the `contrail-vrouter-nodes-8` section completely

d. Similarly, remove the `contrail-vrouter-nodes-14` section completely

e. Add the following `contrail-vrouter-nodes` section to the end:

```
---
apiVersion: dataplane.juniper.net/v1alpha1
kind: Vrouter
metadata:
  name: contrail-vrouter-nodes
  namespace: contrail
spec:
  clusterName: distributed-cluster
  common:
    affinity:
      nodeAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          nodeSelectorTerms:
```

```
            - matchExpressions:
              - key: node-role.kubernetes.io/master
                operator: NotIn
                values:
                - ""
      imagePullSecrets:
        - name: registrypullsecret
      containers:
        - image: hub.juniper.net/cn2/contrail-vrouter-agent:22.1.0.93
          name: contrail-vrouter-agent
        - image: hub.juniper.net/cn2/contrail-init:22.1.0.93
          name: contrail-watcher
        - image: hub.juniper.net/cn2/contrail-telemetry-exporter:22.1.0.93
          name: contrail-vrouter-telemetry-exporter
      initContainers:
        - image: hub.juniper.net/cn2/contrail-init:22.1.0.93
          name: contrail-init
        - image: hub.juniper.net/cn2/contrail-cni-init:22.1.0.93
          name: contrail-cni-init
    maxUnavailablePercentage: 100%
```

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

```
user@workload:~/contrail$ kubectl apply -f manifests/deployer_ha_kernel.yaml
```

After a few minutes, verify that all pods are up:

```
user@workload:~# kubectl get pods -A
NAMESPACE         NAME                                     READY   STATUS      RESTARTS
contrail-deploy   contrail-k8s-deployer-6c88f6798f-jjx96   1/1     Running     3
contrail          contrail-vrouter-masters-phttx          3/3     Running     0
contrail          contrail-vrouter-nodes-xkkk8            3/3     Running     0
kube-system       coredns-657959df74-26jrk                1/1     Running     0
kube-system       dns-autoscaler-b5c786945-m4vtz          1/1     Running     0
kube-system       kube-apiserver-k8s-cp0                  1/1     Running     0
kube-system       kube-controller-manager-k8s-cp0         1/1     Running     0
kube-system       kube-proxy-5rbp4                        1/1     Running     0
kube-system       kube-proxy-kr95n                        1/1     Running     0
kube-system       kube-scheduler-k8s-cp0                  1/1     Running     0
kube-system       nginx-proxy-k8s-worker0                 1/1     Running     0
```

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

Attach a Workload Cluster Running Kernel Mode Data Plane in Release 22.2 or Higher

Use this procedure to create and attach a distributed workload cluster (running a kernel mode data plane) to the central cluster in release 22.2 or higher. 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 **multi-cluster/ distributed_cluster_deployer_example.yaml** and **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 Kubernetes Cluster" on page 91 or use your own methods to create a cluster. The cluster should have the following characteristics:

   - Cluster has no CNI plug-in.

   - Disable Node Local DNS.

   - 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 Contrail 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.

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

   b. 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**.

```
user@workload:~/contrail$ 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.

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

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

```
user@workload:~/contrail$ kubectl apply -f manifests/
distributed_cluster_deployer_example.yaml
```

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

```
user@workload:~/contrail# kubectl get pods -n contrail-deploy
NAME                                      READY   STATUS    RESTARTS   AGE
contrail-k8s-deployer-6458859585-xhwx6    1/1     Running   0          6m
```

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.

```
user@central:~/contrail$ 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**).

```
user@central:~/contrail$ 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
    nodeSelector:
      node-role.kubernetes.io/master: ""
  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>
```

explains the parameters that you need to set.

Table 18: Kubemanager CRD

Parameter	Meaning	Example
name	The name of the custom resource.	kubemanager-cluster1
image	The repository where you pull images	hub.juniper.net/cn2/contrail-k8s-kubemanager:22.1.0.93 enterprise-hub.juniper.net/contrail-container-prod/contrail-k8s-kubemanager:22.3.0.71

Table 18: Kubemanager CRD *(Continued)*

Parameter	Meaning	Example
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
enableNad	True or false (to enable network address definition or not)	true

Table 18: Kubemanager CRD *(Continued)*

Parameter	Meaning	Example
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.

```
user@central:~/contrail$ 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.

```
user@workload:~/contrail$ kubectl apply -f manifests/distributed_cluster_vrouter_example.yaml
```

After a few minutes, verify that all pods are up:

```
user@workload:~# kubectl get pods -A

NAMESPACE          NAME                                      READY   STATUS      RESTARTS

contrail-deploy    contrail-k8s-deployer-6c88f6798f-jjx96    1/1     Running     3

contrail           contrail-vrouter-masters-phttx           3/3     Running     0

contrail           contrail-vrouter-nodes-xkkk8             3/3     Running     0

kube-system        coredns-657959df74-26jrk                 1/1     Running     0

kube-system        dns-autoscaler-b5c786945-m4vtz           1/1     Running     0

kube-system        kube-apiserver-k8s-cp0                   1/1     Running     0

kube-system        kube-controller-manager-k8s-cp0          1/1     Running     0

kube-system        kube-proxy-5rbp4                         1/1     Running     0

kube-system        kube-proxy-kr95n                         1/1     Running     0

kube-system        kube-scheduler-k8s-cp0                   1/1     Running     0

kube-system        nginx-proxy-k8s-worker0                  1/1     Running     0
```

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.

```
user@central:~/contrail$ kubectl delete kubemanager -n contrail <kubemanager-name>
```

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

```
user@workload:~/contrail$kubectl delete vrouter -n contrail contrail-vrouter-masters
```

```
user@workload:~/contrail$ 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.

```
user@central:~/contrail$ kubectl get ns
```

```
user@central:~/contrail$ 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 Contrail

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

1. Follow the steps in "Detach a Workload Cluster" on page 87 to detach the workload cluster that you want to delete.

2. Uninstall Contrail in the workload cluster.

   a. Follow the steps in "Uninstall Contrail" on page 71 to uninstall Contrail 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.

```
user@workload:~/contrail$ kubectl api-resources --verbs=list --namespaced -o name | xargs -
n 1 kubectl get --show-kind --ignore-not-found -n <default pod network namespace>
```

```
user@workload:~/contrail$ 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 Contrail from the workload cluster, then you're done. If you want to also uninstall Contrail from the central cluster, then proceed to the next step.

3. Uninstall Contrail in the central cluster.

   a. Follow the steps in to uninstall Contrail 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.

   ```
   user@central:~/contrail$ kubectl api-resources --verbs=list --namespaced -o name | xargs -
   n 1 kubectl get --show-kind --ignore-not-found -n contrail
   ```

   ```
   user@central:~/contrail$ kubectl api-resources --verbs=list --namespaced -o name | xargs -
   n 1 kubectl get --show-kind --ignore-not-found -n contrail-system
   ```

   ```
   user@central:~/contrail$ 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 Kubernetes Cluster

Use this example procedure to create an upstream Kubernetes cluster.

We provide this example procedure purely for informational purposes. There are multiple ways you can create a cluster, such as with kubeadm, kOps, or kubespray, among others.

In this example, we'll use kubespray and Ansible to create the cluster. Kubespray uses Ansible playbooks, which makes cluster creation fairly straightforward. To make the steps easier to follow, we'll use a separate installer machine to perform the installation and to run kubectl and other tools.

For more information on creating a cluster, see the official Kubernetes documentation (https://kubernetes.io/docs/home/).

> **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.

1. Install a fresh OS on the installer machine, configuring the OS minimally for the following:

   - static IP address and mask (for example, 172.16.0.10/24 for our single cluster) and gateway

   - access to one or more DNS servers

   - SSH connectivity including root SSH access

   - NTP

   The installer machine used in our examples is a Ubuntu host attached to the cluster network.

2. From your local computer, SSH into the installer machine as the sudo user.

3. Install ansible.

   ```
   sudo apt install ansible
   ```

4. Install kubectl. In this example, we run kubectl on the installer machine. If you want to run kubectl on another machine (for example, your local computer), download and install kubectl on that machine instead.

- Set up and update the Kubernetes repository.

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add
```

```
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/
sources.list.d/kubernetes.list
```

```
sudo apt update
```

- Install kubectl.

```
sudo apt install kubectl
```

5. If you want to install Contrail Analytics, then install Helm 3.0 or later.

- Set up and update the Helm repository.

```
curl https://baltocdn.com/helm/signing.asc | sudo apt-key add -
```

```
echo "deb https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/
sources.list.d/helm-stable-debian.list
```

```
sudo apt update
```

- Install Helm.

```
sudo apt install helm
```

6. Configure SSH password-less root access from the installer machine to the control plane and worker nodes. This allows ansible to log in to these nodes when you run the playbook later.

a. Create an SSH key.

```
user@installer:~$ ssh-keygen
```

In this example, we store the SSH key in its default location **~/.ssh/id_rsa.pub**.

b. Copy the key to the root user on the control plane and worker nodes. For example:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub root@172.16.0.11
ssh-copy-id -i ~/.ssh/id_rsa.pub root@172.16.0.12
ssh-copy-id -i ~/.ssh/id_rsa.pub root@172.16.0.13
```

7. Clone the kubespray directory.

For example:

```
user@installer:~/contrail$ git clone https://github.com/kubernetes-sigs/kubespray -b
release-2.16
```

This creates a clone of the kubespray directory at the present location (in **~/contrail** in this example).

8. Configure the pod and service subnets if desired.

The default subnets used by kubespray are defined in the **kubespray/roles/kubespray-defaults/defaults/main.yaml** file. Look for the following parameters in that file and change accordingly.

```
kube_service_addresses: 10.233.0.0/18
kube_pods_subnet: 10.233.64.0/18
kube_service_addresses_ipv6: fd85:ee78:d8a6:8607::1000/116
kube_pods_subnet_ipv6: fd85:ee78:d8a6:8607::1:0000/112
```

NOTE: If you're creating a multi-cluster Contrail setup, you must configure different pod and service subnets on each cluster. These subnets must be unique within the entire multi-cluster.

If you're following the multi-cluster example in this document, then leave the subnets on the central cluster at their defaults and configure the subnets on the workload cluster as follows:

```
kube_service_addresses: 10.234.0.0/18
kube_pods_subnet: 10.234.64.0/18
kube_service_addresses_ipv6: fd85:ee78:d8a6:8608::1000/116
kube_pods_subnet_ipv6: fd85:ee78:d8a6:8608::1:0000/112
```

9. Disable Node Local DNS.

In the **kubespray/roles/kubespray-defaults/defaults/main.yaml** file, set `enable_nodelocaldns: false`.

10. If you're running DPDK in your cluster, then configure multus.

    Multus is required when running DPDK.

    a. Enable multus.

       In **kubespray/roles/kubespray-defaults/defaults/main.yaml**, enable multus:

       ```
       kube_network_plugin_multus: true
       ```

    b. Set the multus version to 0.3.1, which is the version required for running DPDK. You set the version in two files.

       In **kubespray/roles/network_plugin/multus/defaults/main.yml**, configure the multus version:

       ```
       multus_cni_version: "0.3.1"
       ```

       In **kubespray/extra_playbooks/roles/network_plugin/multus/defaults/main.yml**, configure the multus version:

       ```
       multus_cni_version: "0.3.1"
       ```

11. Create the inventory file for ansible to use. For example:

    ```
    all:
        hosts:                                  # list all nodes here
            k8s-cp0:                            # desired hostname
                ansible_host: 172.16.0.11
            k8s-worker0:
                ansible_host: 172.16.0.12
            k8s-worker1:
                ansible_host: 172.16.0.13
        vars:
            ansible_user: root
            artifacts_dir: /tmp/mycluster
            cluster_name: mycluster.contrail.lan
            container_manager: crio             # container runtime
            docker_image_repo: <your docker repository URL>
            download_container: false
            download_localhost: true
            download_run_once: true
            enable_dual_stack_networks: true
    ```

```
        enable_nodelocaldns: false
        etcd_deployment_type: host
        host_key_checking: false
        kube_network_plugin: cni
        kube_network_plugin_multus: false
        kubeconfig_localhost: true
        kubectl_localhost: true
        kubelet_deployment_type: host
        override_system_hostname: true
  kube-master:
      hosts:                     # hostname of control plane node (from hosts section)
        k8s-cp0:
  kube-node:
      hosts:                     # hostnames of worker nodes (from hosts section)
        k8s-worker0:
        k8s-worker1:
  etcd:
      hosts:                     # hostname of control plane node (from hosts section)
        k8s-cp0:
  k8s-cluster:
      children:
        kube-master:
        kube-node:
```

The host names (`k8s-cp0`, `k8s-worker0`, `k8s-worker1`) that you specify in the file are automatically configured on the node when the `override_system_hostname` parameter is set to `true`.

NOTE: If you're creating a multi-cluster Contrail setup, you must configure different node names for each node in the multi-cluster. Node names must be unique across the entire multi-cluster.

NOTE: If you're running DPDK, set `kube_network_plugin_multus: true`.

If you want to run with a different container runtime, change the `container_manager` value above.

Ensure `enable_nodelocaldns` is set to `false`.

If you want to run with a different number of control plane and worker nodes, adjust the inventory accordingly.

12. Check that ansible can SSH into the control plane and worker nodes based on the contents of the **inventory.yaml** file. In this example, the **inventory.yaml** file is in the **~/contrail** directory.

```
user@installer:~/contrail$ ansible -i inventory.yaml -m ping all
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see
details
k8s-cp0 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
k8s-worker0 | SUCCESS => {ansi
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
k8s-worker1 | SUCCESS => {ansi
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

13. To create the cluster, run the playbook from the **kubespray** directory. Adjust the command below to reference the **inventory.yaml** within your directory structure.

```
user@installer:~/contrail/kubespray$ ansible-playbook -i ../inventory.yaml cluster.yml
```

This step can take a an hour or more to complete, depending on the size of your cluster.

> **NOTE**: You can safely ignore network and CNI warnings and errors because you haven't configured a CNI yet. If a fatal error occurs, ansible will stop the playbook.

14. Copy the cluster's secure token to the default **~/.kube/config** location. The kubeconfig must be at that default location for Contrail tools to work.

You can find the secure token location from the **inventory.yaml** file. If you use the inventory file in this example, the token is in **/tmp/mycluster**.

```
mkdir ~/.kube
```

```
cp /tmp/mycluster/admin.conf ~/.kube/config
```

NOTE: If you have a kubeconfig that already holds tokens for existing clusters, then you'll need to merge rather than overwrite the **~/.kube/config** file.

15. Use standard kubectl commands to check on the health of the cluster.

a. Show the status of the nodes.

```
user@installer:~$ kubectl get nodes

NAME          STATUS     ROLES                 AGE     VERSION

k8s-cp0       NotReady   control-plane,master  4m57s   v1.20.7

k8s-worker0   NotReady   <none>                3m17s   v1.20.7

k8s-worker1   NotReady   <none>                2m45s   v1.20.7


user@installer:~$ kubectl describe node k8s-cp0

<trimmed>

Conditions:

  Type           Status    <trimmed>    Reason                    Message

  ----           ------                 ------                    -------

  MemoryPressure False                  KubeletHasSufficientMemory kubelet has sufficient memory available

  DiskPressure   False                  KubeletHasNoDiskPressure   kubelet has no disk pressure

  PIDPressure    False                  KubeletHasSufficientPID    kubelet has sufficient PID available

  Ready          False                  KubeletNotReady            runtime network not ready: NetworkReady=false

  reason:NetworkPluginNotReady message:Network plugin returns error: No CNI configuration file in /etc/cni/net.d/. Has your

  network

  Addresses:

  <trimmed>
```

You can see that the nodes are not ready because there is no CNI plug-in. This is expected because you haven't installed Contrail yet.

b. Show the status of the pods.

```
user@installer:~$ kubectl get pods -A -o wide
NAMESPACE     NAME                        READY  STATUS    RESTARTS  AGE    IP           NODE         <trimmed>
kube-system   coredns-657959df74-rprzv    0/1    Pending   0         6m44s  <none>       <none>       <none>      <none>
kube-system   dns-autoscaler-b5c786945    0/1    Pending   0         6m40s  <none>       <none>       <none>      <none>
kube-system   kube-apiserver-k8s-cp0      1/1    Running   0         9m27s  172.16.0.11  k8s-cp0      <none>      <none>
kube-system   kube-controller-manager-    1/1    Running   0         9m27s  172.16.0.11  k8s-cp0      <none>      <none>
kube-system   kube-proxy-k5mcp            1/1    Running   0         7m28s  172.16.0.13  k8s-worker1  <none>      <none>
kube-system   kube-proxy-sccjm            1/1    Running   0         7m28s  172.16.0.11  k8s-cp0      <none>      <none>
kube-system   kube-proxy-wqbt8            1/1    Running   0         7m28s  172.16.0.12  k8s-worker0  <none>      <none>
kube-system   kube-scheduler-k8s-cp0      1/1    Running   0         9m27s  172.16.0.11  k8s-cp0      <none>      <none>
kube-system   nginx-proxy-k8s-worker0     1/1    Running   0         8m2s   172.16.0.12  k8s-worker0  <none>      <none>
kube-system   nginx-proxy-k8s-worker1     1/1    Running   0         7m30s  172.16.0.13  k8s-worker1  <none>      <none>
```

All pods should have a STATUS of Running except for the DNS pods. The DNS pods do not come up because there is no networking. This is what we expect.

You have successfully installed a new cluster. You can now proceed to install Contrail on this cluster.

Run Preflight and Postflight Checks

SUMMARY

Learn about Contrail preflight and postflight checks.

IN THIS SECTION

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

Postflight checks allow you to verify that your Contrail 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 Contrail. Contrail supports postflight checks starting in release 22.2.

Run Preflight Checks in Release 22.1

In Release 22.1, we provide a preflight check script that you run on each node in the cluster. The script checks to see if the cluster node is able to run the deployer manifest that you want to apply to that node.

1. Locate the **preflight.sh** script and the manifest that you want to apply from the downloaded package.

2. Copy the script and the manifest(s) you plan on using to every node in the cluster. You're not applying the manifest yet. You're just making the manifest available for the **preflight.sh** script to read.

3. Log in as root to each node in the cluster and run the script.

```
root@node:~$ ls
<deployer-manifest>.yaml
preflight.sh
```

where ***<deployer-manifest>*.yaml** is the name of the manifest(s) that you plan to apply to this node.

```
root@node:~$ ./preflight.sh -d .
```

The script checks to make sure that the node is capable of supporting the resources required by the ***<deployer-manifest>*.yaml** file.

Address any errors flagged by the script before proceeding.

Run Preflight and Postflight Checks in Release 22.2

In release 22.2, we provide a custom controller that performs preflight and postflight checks on all cluster nodes. The controller runs the checks defined in custom resources that we provide. You create and run the controller in the same way you run other Kubernetes applications. You no longer need to log in to each cluster node to run the checks.

1. Locate the **contrail-tools/contrail-readiness** directory from the downloaded Contrail Manifests and 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 103 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 want to apply to 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.

5. Create the ContrailReadiness controller.

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

Wait for the controller to come up.

6. Run the checks.

- To run the preflight checks:

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

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

- To run the postflight checks:

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

You run postflight checks after you install Contrail.

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

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

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

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

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

```
kubectl get contrailreadinesstest 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.

Run Preflight and Postflight Checks in Release 22.3

In release 22.3, we provide a custom controller that performs preflight and postflight checks on all cluster nodes. The controller runs the checks defined in custom resources that we provide. You create and run the controller in the same way you run other Kubernetes applications.

1. Locate the **contrail-tools/contrail-readiness** directory from the downloaded Contrail Manifests and 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 103 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 want to apply to 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.

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
```

Wait for the controller to come up.

7. Run the checks.

- To run the preflight checks:

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

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

- To run the postflight checks:

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

You run postflight checks after you install Contrail.

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

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

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

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

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

```
kubectl get contrailreadinesstest 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.

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.

   In releases 22.1 and 22.2, it's hub.juniper.net:

   ```
   docker login hub.juniper.net
   ```

   In releases 22.3 and higher, it's enterprise-hub.juniper.net:

   ```
   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)
```

4. Replace the credentials placeholder in the manifests with the encoded string.

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

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

RELATED DOCUMENTATION

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

Add a Cluster Node

Use this procedure to add a control plane or worker node to a cluster that you used kubespray to create.

We provide this example procedure purely for informational purposes. See Kubernetes documentation (https://kubernetes.io/docs/home/) for the official procedure.

1. Prepare the new node by following step 5 in "Before You Install" on page 18.
2. Add the node to the inventory.yaml file that you created in step 11 in "Create a Kubernetes Cluster" on page 91.
3. Verify that ansible can SSH into all the nodes.

```
ansible -i inventory.yaml -m ping all
```

4. Run the playbook from the **kubespray** directory. Adjust the command below to reference the **inventory.yaml** within your directory structure.

```
user@installer:~/contrail/kubespray$ ansible-playbook -i ../inventory.yaml cluster.yml
```

Contrail automatically detects that you've add a new node and configures new networking components as necessary. You don't have to explicitly configure Contrail for the new node.

5. Use standard kubectl commands to check that the new node is now part of the cluster.

```
kubectl get nodes
```

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

Delete a Cluster Node

Use this procedure to delete a node gracefully.

We provide this example procedure purely for informational purposes. See Kubernetes documentation (https://kubernetes.io/docs/home/) for the official procedure.

1. Drain the node.

```
kubectl drain <node> --ignore-daemonsets --delete-local-data
```

2. Ensure there are no jobs running or scheduled to run on that node.

```
kubectl get jobs -A
```

```
kubectl get cronjobs -A
```

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

3. Delete the node.

```
kubectl delete <node>
```

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.

Back Up and Restore

SUMMARY

Learn how to use etcdctl commands to back up and restore the etcd database.

IN THIS SECTION

We provide these example procedures purely for informational purposes. For more information on back up and restore, see the official Kubernetes documentation (https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/#backing-up-an-etcd-cluster and https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/#restoring-an-etcd-cluster).

Back Up the Etcd Database

Use this example procedure to back up the etcd database.

1. SSH into one of the control plane nodes in the cluster.
2. Install etcdctl version 3.4.13 or later. Etcdctl is the command line tool for managing etcd.

   If the node already has etcdctl version 3.4.13 or later installed, then you can skip this step.

   Otherwise, install etcdctl:

   ```
   root@cp1:~# etcdctl version
   Command 'etcdctl' not found
   ```

   ```
   curl -L https://storage.googleapis.com/etcd/v3.5.0/etcd-v3.5.0-linux-amd64.tar.gz
   ```

   In this example, we move the file to the **/tmp** directory.

   ```
   mv etcd-v3.5.0-linux-amd64.tar.gz /tmp
   ```

Extract and copy the executable to **/usr/local/bin**.

```
mkdir -p /tmp/etcd-download
tar -xzvf /tmp/etcd-v3.5.0-linux-amd64.tar.gz -C /tmp/etcd-download --strip-components=1
```

```
cp /tmp/etcd-download/etcdctl /usr/local/bin
```

Verify by querying the version.

```
root@cp1:~# etcdctl version
etcdctl version: 3.5.0
API version: 3.5
```

3. Set the required ETCDCTL env variables.

```
export ETCDCTL_CACERT=/etc/kubernetes/pki/etcd/ca.crt
export ETCDCTL_CERT=/etc/kubernetes/pki/etcd/server.crt
export ETCDCTL_KEY=/etc/kubernetes/pki/etcd/server.key
export ETCDCTL_API=3
```

These variables are used implicitly by the etcdctl commands. The file paths listed are the default file paths. You can obtain these file paths by issuing the `kubectl describe pod` `<etcd-pod>` command.

4. Set permissions on the certificate files.

```
chmod 777 /etc/kubernetes/pki/etcd/server.key
chmod 777 /etc/kubernetes/pki/etcd/ca.crt
```

5. Repeat step 1 to step 4 on all the control plane nodes.
6. Back up the etcd database.

   SSH back into one of the control plane nodes and take a snapshot of the etcd database.

```
etcdctl snapshot save /tmp/etcdBackup.db --endpoints=https://127.0.0.1:2379
```

   This takes a snapshot of the database and stores it in **/tmp/etcdBackup.db**.

7. Copy the snapshot off the node and store in a safe place.

Restore the Etcd Database

Use this example procedure to restore the etcd database from a snapshot.

1. Restore the snapshot on all the control plane nodes.

   a. SSH into one of the control plane nodes.

   b. Copy the saved snapshot to **/tmp/etcdBackup.db** (for example).

   c. Restore the backup.

   ```
   etcdctl snapshot restore /tmp/etcdBackup.db \
   --name=<cp1-etcd-pod> \
   --initial-cluster=<cp1-etcd-pod>=https://<cp1-etcd-pod-ip>:2380,<cp2-etcd-pod>=https://
   <cp2-etcd-pod-ip>:2380,<cp3-etcd-pod-name>=https://<cp3-etcd-pod-ip>:2380 \
   --initial-advertise-peer-urls= https://<cp1-etcd-pod-ip>:2380
   ```

   where *<cp1-etcd-pod>* is the name of the contrail-etcd pod that you're currently in and *<cp1-etcd-pod-ip>* is the IP address of that pod. The *<cp2-etcd-pod>* and *<cp3-etcd-pod>* refer to the other contrail-etcd pods. This creates a ***<cp1-etcd-pod>*.etcd** directory on the node.

   d. Repeat for the other control plane nodes, substituting the `--name` and `--initial-advertise-peer-urls` values with the respective pod name and IP address.

2. Stop the API server on all the control plane nodes.

   a. SSH into one of the control plane nodes.

   b. Stop the API server.

   ```
   mkdir -p /tmp/k8s
   ```

   ```
   mv /etc/kubernetes/manifests/*.yaml /tmp/k8s
   ```

   c. Repeat for the other control plane nodes.

3. Move the restored etcd snapshot to */var/lib/etcd* on all the control plane nodes.

   a. SSH into one of the control plane nodes.

b. Move the restored etcd snapshot.

```
mv /var/lib/etcd/member /var/lib/etcd/member.bak
```

```
mv <restored-etcd-directory>/member /var/lib/etcd/
```

where *<restored-etcd-directory>* is the **.etcd** directory created in step 1.

c. Repeat for the other control plane nodes.

4. Restore the API server on all control plane nodes.

a. SSH into one of the control plane nodes.

b. Restore the API server.

```
mv /tmp/k8s/*.yaml /etc/kubernetes/manifests
```

c. Repeat for the other control plane nodes.

5. Restart the kubelet on all control plane nodes.

a. SSH into one of the control plane nodes.

b. Restart the kubelet.

```
systemctl stop kubelet
```

```
systemctl start kubelet
```

c. Repeat for the other control plane nodes.

6. Restart the kube-system apiserver and controller.

Delete all the kube-apiserver and kube-controller pods.

```
kubectl delete pod <kube-apiserver-xxx> -n kube-system
```

```
kubectl delete pod <kube-controller-xxx> -n kube-system
```

These pods will automatically restart.

7. Restart the contrail-system apiserver and controller.

Delete all the contrail-k8s-apiserver and contrail-k8s-controller pods.

```
kubectl delete pod <contrail-k8s-apiserver-xxx> -n contrail-system
```

```
kubectl delete pod <contrail-k8s-controller-xxx> -n contrail-system
```

These pods will automatically restart.

8. Restart the vrouters.

Delete all the contrail-vrouter-masters and contrail-vrouter-nodes pods.

```
kubectl delete pod <contrail-vrouter-masters-xxx> -n contrail
```

```
kubectl delete pod <contrail-vrouter-nodes-xxx> -n contrail
```

These pods will automatically restart.

9. Check that all pods are in running state.

```
kubectl get pods -n contrail-system
```

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

Juniper CN2 Technology Previews (Tech Previews)

Tech Previews give you the ability 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.