Juniper Networks | Engineering Simplicity

Cloud Native Contrail Networking

Installation and Life Cycle Management Guide for Amazon EKS

Published
2023-09-08

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

Cloud Native Contrail Networking Installation and Life Cycle Management Guide for Amazon EKS

The information in this document is current as of the date on the title page.

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

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at https://support.juniper.net/support/eula/. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

5

Appendix

CHAPTER 1

Introduction

Cloud-Native Contrail Networking Overview

SUMMARY

Learn about Cloud-Native Contrail Networking (CN2).

IN THIS SECTION

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

Unless otherwise indicated, all references to Kubernetes 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 (CN2) brings this rich SDN feature set natively to Kubernetes as a networking platform and container network interface (CNI) plug-in.

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

- Manage CN2 using standard Kubernetes and third-party tools.
- Scale CN2 by adding or removing nodes.
- Configure CN2 by using custom resource definitions (CRDs).
- Upgrade CN2 software by applying updated manifests.
- Uninstall CN2 by deleting Contrail namespaces and resources (upstream Kubernetes only).

More than a CNI plug-in, CN2 is a networking platform that provides dynamic end-to-end virtual networking and security for cloud-native containerized and virtual machine (VM) workloads, across multi-cluster compute and storage environments, all from a central point of control. It supports hard

multi-tenancy for single or multi-cluster environments shared across many tenants, teams, applications, or engineering phases, scaling to thousands of nodes.

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

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

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

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

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

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

Benefits of Cloud-Native Contrail Networking

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

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 strictly 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. CN2 supports a centralized network control plane architecture where the routing daemon runs centrally within the Contrail controller and learns and distributes routes from and to the data plane components. This centralized architecture facilitates virtual network abstraction, orchestration, and automation.
Network configuration plane	The network configuration plane interacts with Kubernetes control plane components to manage all CN2 resources. You configure CN2 resources using custom resource definitions (CRDs).
Network data plane	The network data plane resides on all nodes and interacts with containerized workloads to send and receive network traffic. Its main component is the Contrail vRouter.
Contrail controller	This is the part of CN2 that provides the network configuration and network control plane functionality. The Contrail controller typically resides on a Kubernetes control plane node but can reside on a worker node in some cases. 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.

CN2 Components

NOTE: This section is intended to provide a brief and general overview of the components that make up the Cloud-Native Contrail Networking (CN2) solution. These components are generally

common to all Kubernetes distributions running CN2. Nevertheless, differences do exist, and these are called out explicitly where necessary.

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

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

The pods that perform the configuration and control plane functions reside on Kubernetes control plane nodes in most distributions. The pods that perform the data plane functions reside on both Kubernetes control plane nodes and Kubernetes worker nodes. (In Amazon EKS, all CN2 pods reside on worker nodes.)

Table 2 on page 6 describes the main CN2 components. Depending on configuration, there might be other components as well (not shown) that perform ancillary functions such as certificate management and status monitoring.

Table 2: CN2 Components

Pod Name		Where	Description
Configuration Plane[1]	contrail-k8s-apiserver	Control Plane Node[2]	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.

Table 2: CN2 Components *(Continued)*

Pod Name		Where	Description
Control Plane[1]	contrail-k8s-controller	Control Plane Node[2]	This pod performs the Kubernetes control loop function to reconcile networking resources. It constantly monitors networking resources to make sure the actual state of a resource matches its intended state. There is one contrail-k8s-controller pod per Kubernetes control plane node.
	contrail-k8s-kubemanager	Control Plane Node[2]	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.
	contrail-control	Control Plane Node[2]	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: CN2 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[3]	Control Plane Node	This pod provides the same functionality as the contrail-vrouter-nodes pod, but resides on the control plane nodes.

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

[2]Worker node if running on Amazon EKS.

[3]Not present if running on Amazon EKS.

Figure 1 on page 9 shows these components in the context of a Kubernetes cluster in most distributions and Figure 2 on page 10 shows these components in the context of Amazon EKS.

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

Figure 1: CN2 Components (Most Distributions)

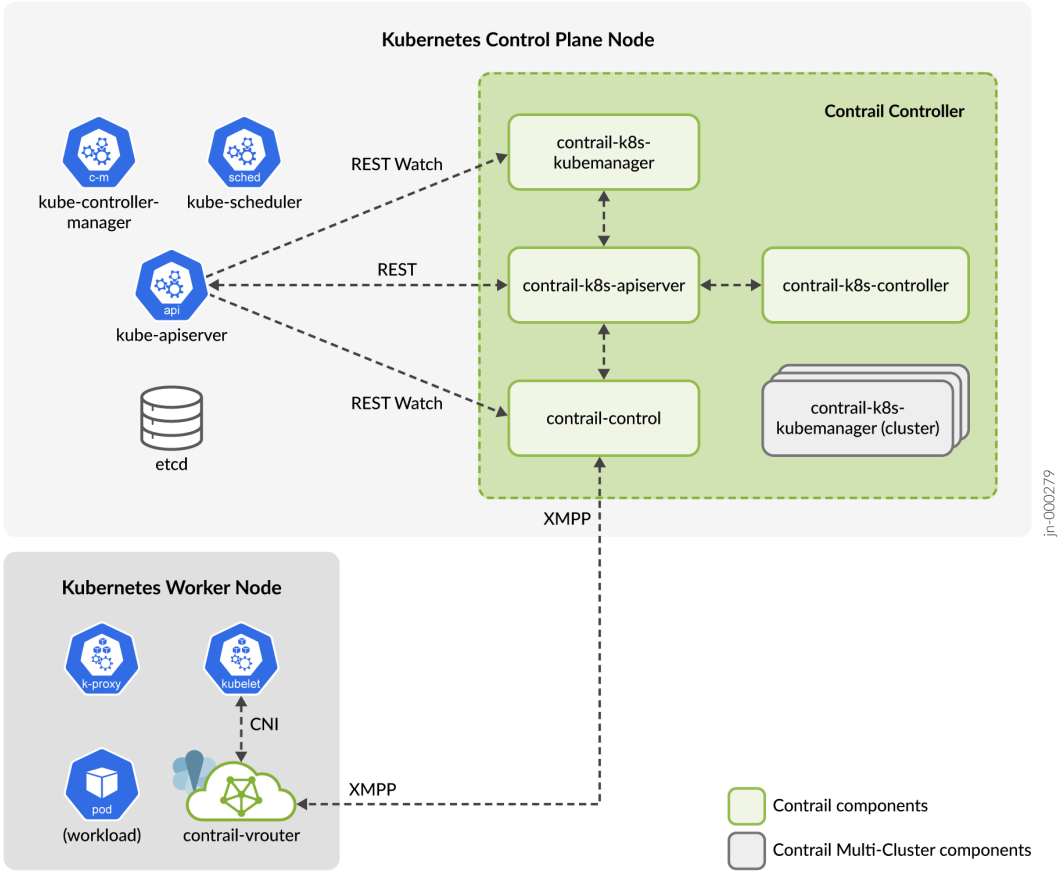
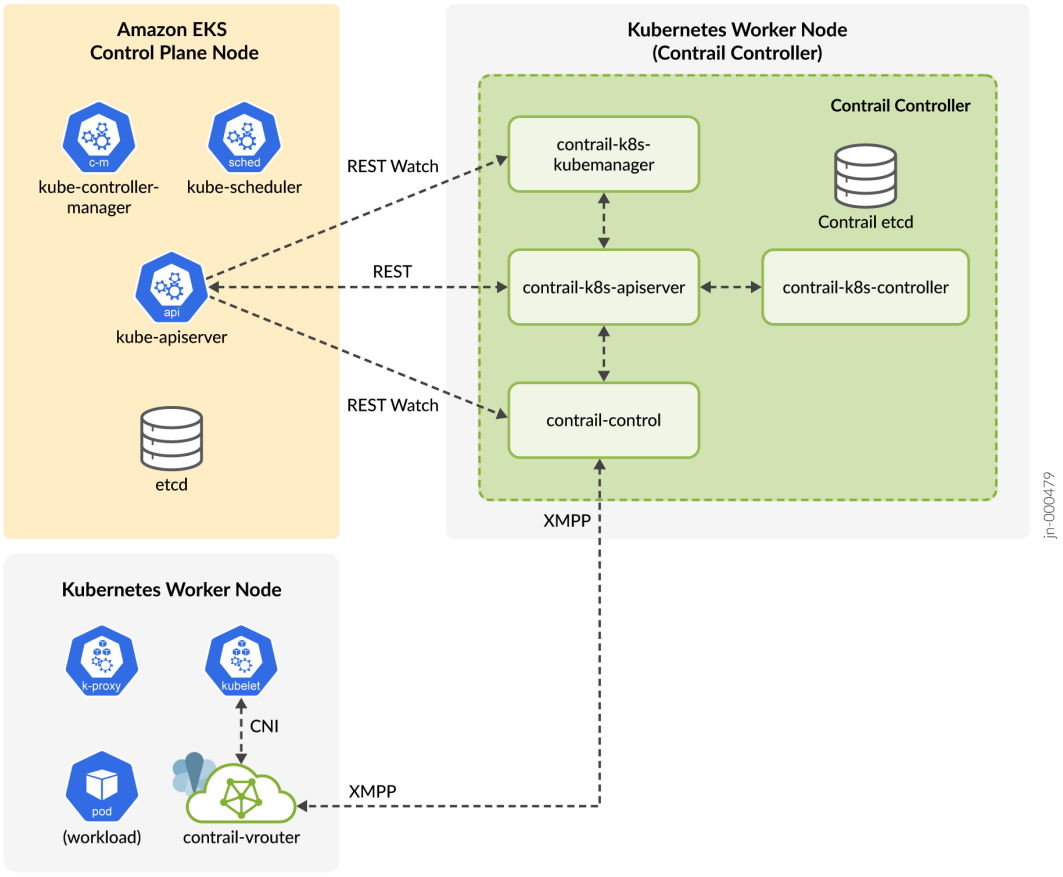


Figure 2: CN2 Components (Amazon EKS)



When running on upstream Kubernetes, CN2 uses the main Kubernetes etcd database. When running on OpenShift or when running on Amazon EKS, CN2 uses its own etcd database.

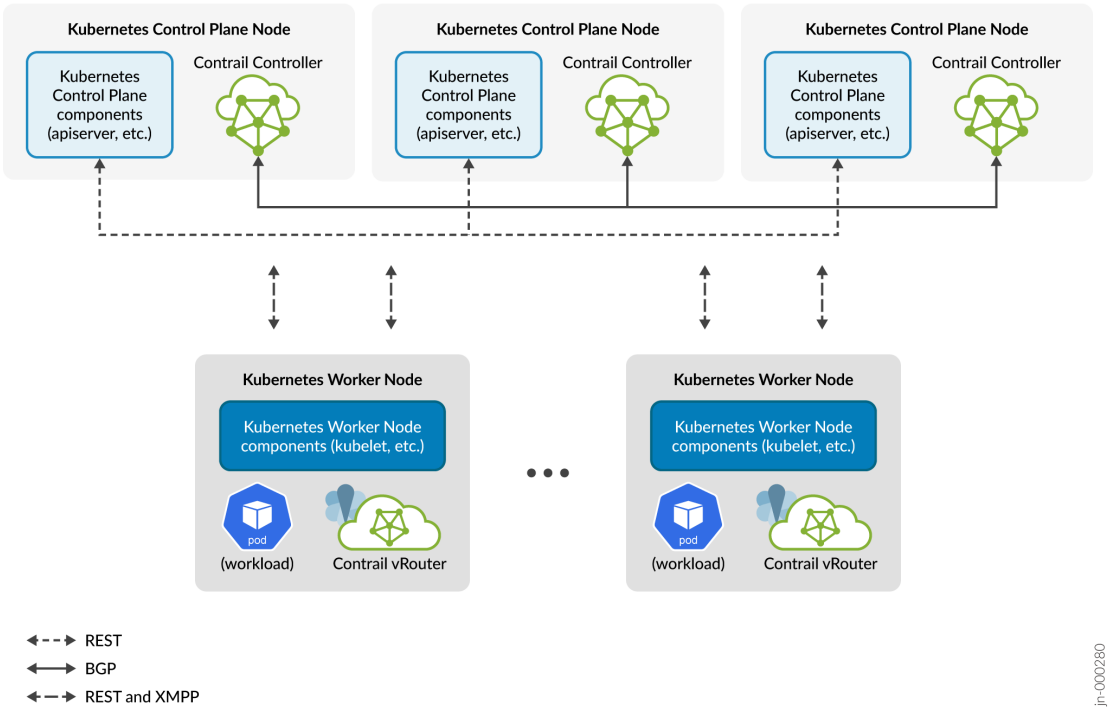
For all distributions, the kube-apiserver is the entry point for Kubernetes REST API calls for the cluster. It directs all networking requests to the contrail-k8s-apiserver, which is the entry point for Contrail API calls. The contrail-k8s-apiserver translates incoming networking requests into REST API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

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

Figure 3 on page 11 shows a cluster with multiple Contrail controllers. These controllers reside on control plane nodes (most distributions). The Kubernetes components communicate with each other using REST. The Contrail controllers exchange routes with each other using iBGP, outside of the regular

Kubernetes REST interface. For redundancy, the vRouter agents on worker nodes always establish XMPP communications with two Contrail controllers.

Figure 3: Multiple Contrail Controllers (Most Distributions)



Deployment Models

SUMMARY

Learn about the different ways you can deploy CN2 on Amazon EKS.

IN THIS SECTION

Single Cluster Deployment

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

In a single-cluster deployment on Amazon EKS, the Contrail controller sits in chosen worker nodes and provides the network configuration and network control planes for the host cluster. The Contrail data plane components sit in all worker nodes and provide the packet send and receive function for the workloads.

Multi-Cluster Deployment

In a multi-cluster deployment, the Contrail controller resides in its own cluster and provides networking to other clusters.

CN2 does not support multi-cluster deployment on an Amazon EKS cluster.

System Requirements

Table 3: System Requirements for Amazon EKS Installation with CN2

Machine	Instance Type	Notes
Control Plane Nodes	Not applicable	Outside of user control. Automatically scales when needed.
Worker Nodes for CN2	m5.xlarge	For worker nodes running the Contrail controller and Contrail data plane only (no user workloads).
Worker Nodes	m5.xlarge[1]	For worker nodes running user workloads and the Contrail data plane (no Contrail controller).
[1]This is a suggestion only. Actual compute requirements depend on the actual workoads.		

2
CHAPTER

Install

Overview

IN THIS SECTION

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed cloud-based Kubernetes service that provides the Kubernetes infrastructure for your applications. Amazon EKS manages the performance, scale, reliability, and availability of the cluster automatically without requiring your intervention. With Amazon EKS, you're only responsible for managing the worker nodes where your Kubernetes applications are running. You don't need to install, operate, or maintain the Kubernetes control plane, which is completely hidden from you.

Amazon EKS runs in the AWS cloud, with the Kubernetes control plane residing in an Amazon-provided Virtual Private Cloud (VPC) and the worker nodes running under your control in your own VPC. The control plane nodes span multiple AWS Availability Zones and automatically scale to adjust to load and health.

When augmented with CN2, Amazon EKS gains a full-featured CNI and networking platform that can meet the complex networking requirements of small and large businesses alike. CN2 not only provides pod-to-pod networking for the Amazon EKS cluster, but it simplifies hybrid cloud deployments with secure network segmentation and seamless connectivity from the private edge to the public core.

Benefits of Amazon EKS with CN2

- Industry-leading managed Kubernetes service together with industry-leading SDN
- Full featured SDN solution for all types of enterprises
- Seamless end-to-end networking in hybrid cloud environments regardless of geography
- Centralized SDN control with elastic performance and scale

Before You Install

1. Set up an account with AWS.

   You'll need the AWS account to set up an Amazon EKS cluster.
2. Install the AWS CLI. See https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html.
3. Install kubectl. See https://kubernetes.io/docs/tasks/tools/.
4. Install Terraform. See https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli.
5. Enable AWS account permissions for running Terraform. See https://developer.hashicorp.com/terraform/tutorials/aws-get-started/aws-build.

   The policy resource set in `terraform-aws-eks-blueprints/examples/eks-cluster-with-cn2/min-iam-policy.json` allows all resources. We recommend you change this in a real deployment.
6. Optionally, install contrailstatus. Contrailstatus is a kubectl plug-in you can use to query CN2 microservices and CN2-specific resources.

   The contrailstatus executable is available from GitHub ("Tools" on page 19).

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

Install CN2 on Amazon EKS

SUMMARY

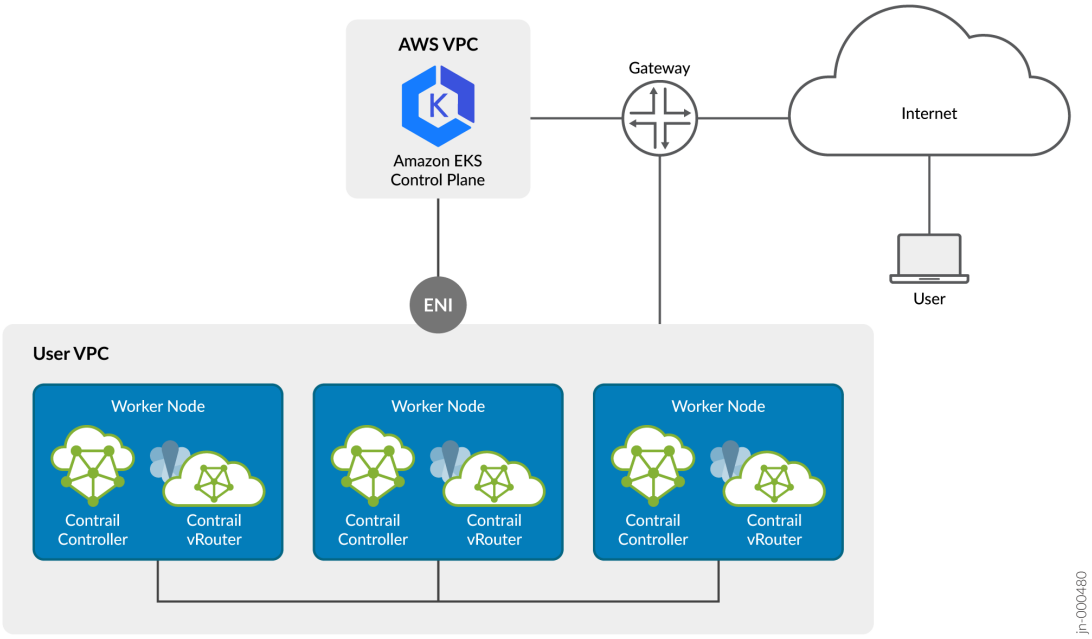
See examples on how to install single cluster CN2 on Amazon EKS.

IN THIS SECTION

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

Figure 4 on page 16 shows a cluster of three worker nodes running the Contrail controller in an Amazon EKS cluster. The control plane nodes are managed by AWS and are not under user control.

Figure 4: CN2 on Amazon EKS



All communication between nodes in the cluster and between nodes and external sites takes place over the AWS network, in the same manner as a standard Amazon EKS cluster.

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

Table 4: Single Cluster Examples

Release	Kernel Mode Data Plane	DPDK Data Plane
22.4	"Install Single Cluster CN2 Running Kernel Mode Data Plane in Release 22.4" on page 17	Not supported
NOTE: The provided manifests might not be compatible between releases. Make sure you use the manifests for the release that you're running. In practice, this means that you should not modify the image tag in the supplied manifests.		

Install Single Cluster CN2 Running Kernel Mode Data Plane in Release 22.4

Use this procedure to install CN2 in an Amazon EKS cluster running a kernel mode data plane in release 22.4.

This example procedure uses Terraform to deploy the following basic Amazon EKS Cluster with VPC:

- creates a new sample VPC, 3 private subnets, and 3 public subnets
- creates Internet gateway for public subnets and NAT gateway for private subnets
- creates EKS Cluster control plane with one managed node group (desired nodes set to 3)
- deploys CN2 as Amazon EKS cluster CNI

1. Clone the AWS Integration and Automation repository. This is where the Terraform manifests are stored.

   ```
   git clone https://github.com/Juniper/terraform-aws-eks-blueprints.git
   ```

2. Add your enterprise-hub.juniper.net access credentials to **terraform-aws-eks-blueprints/examples/eks-cluster-with-cn2/main.tf** .

   The credentials that you add must be base64-encoded. See "Configure Repository Credentials" on page 37 for an example of how to obtain and encode your credentials, and apply it to this Terraform manifest.

3. Run `terraform init`. This command initializes a working directory containing Terraform configuration files.

   ```
   cd examples/eks-cluster-with-cn2
   ```

   ```
   terraform init
   ```

4. Run `terraform plan`. This command creates an execution plan, which lets you preview the changes that Terraform plans to make to your infrastructure.

```
export AWS_REGION=<ENTER YOUR REGION>   # Select your own region
```

```
terraform plan
```

Verify the resources created by this execution.

5. Run `terraform plan`. This command executes the Terraform plan you just created.

```
terraform apply
```

Enter `yes` to apply and create the cluster.

6. Obtain the cluster name and other details of your new Amazon EKS cluster from the Terraform output or from the AWS Console.
7. Copy the kubeconfig onto your local computer.

```
aws eks --region <enter-your-region> update-kubeconfig --name <cluster-name>
```

8. Check over your new cluster.

   List your worker nodes:

```
kubect get nodes
```

List all the pods:

```
kubectl get pods -A
```

9. Remove the aws-node daemonset. This daemonset installs vpc-cni on the worker nodes, which is not needed for this procedure.

```
kubectl -n kube-system delete ds aws-node
```

10. Log in to each worker node to remove the `/etc/cni/net.d/10-aws.conflist` file and reboot.

    For information on how to log in to a node, see https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-connect-methods.html.

Tools

SUMMARY

We provide tools to help you manage your CN2 installation. You can download these tools from GitHub.

IN THIS SECTION

Tools in Release 22.4

IN THIS SECTION

CN2 tools are available for download from GitHub (https://github.com/Juniper/contrail-networking/tree/main/releases/22.4/contrail-tools).

NOTE: We do not provide YAMLs for CN2 installation because the current release of CN2 supports installation using Terraform only.

The following table lists miscellaneous tools that we provide.

Table 5: Tools for Amazon EKS for Release 22.4

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

Table 5: Tools for Amazon EKS for Release 22.4 *(Continued)*

Tools	Description
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.tar.gz	Not applicable for Amazon EKS installations
contrail-tools/kubectl-contrailstatus-tar.gz	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. Contrail Analytics is compatible with CN2 within the same release only.

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

The release 22.4 package is called **contrail-analytics-22.4.0.284.tgz**. See "Install Contrail Analytics" on page 22.

3

CHAPTER

Monitor

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 CN2 telemetry exporters to provide an industry-standard way for you to monitor and analyze your network and network infrastructure. Information collected includes logs, metrics, status' of various component, and flows.

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 22.4

Use this procedure to install Contail Analytics in release 22.4 on Amazon EKS.

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. This can be any IP address as long as it's externally reachable from outside the cluster.

Alternatively, you can specify an Internet-facing AWS load balancer service by adding `--set enableLoadBalancer=true` instead of `--set externalIP=<ipAddress>`. You can then find the external load balancer IP address by `kubectl get service -n contrail-analytics ambassador-loadbalancer` and look for the `EXTERNAL-IP` 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 40 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. This can be any IP address as long as it's externally reachable from outside the cluster.

   Alternatively, you can specify an Internet-facing AWS load balancer service by adding `--set enableLoadBalancer=true` instead of `--set externalIP=<ipAddress>`. You can then find the external load balancer IP address by `kubectl get service -n contrail-analytics ambassador-loadbalancer` and look for the `EXTERNAL-IP` 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 cresource all [detail]
kubectl contrailstatus configdump
kubectl contrailstatus --all [--wide]
kubectl contrailstatus version
```

Description

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

Options

kubectl contrailstatus deployment --plane config	Displays the status of the Configuration plane components: • contrail-k8s-apiserver • contrail-k8s-controller • contrail-k8s-kubemanager
kubectl contrailstatus deployment --plane control	Displays the status of the Control plane components: • contrail-control

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

Additional Information

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

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

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

Output Fields

Table 6 on page 27 lists some of the output fields for the `kubectl contrailstatus` command.

Table 6: kubectl contrailstatus Output Fields

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

Sample Output

kubectl contrailstatus --all

```
user@host> kubectl contrailstatus --all
```

```
NAME(CONFIG)                                STATUS  NODE    IP             MESSAGE
contrail-k8s-apiserver-6d79c8598d-8lfnm     ok      ocp1    172.16.0.11
contrail-k8s-apiserver-6d79c8598d-q7klk     ok      ocp3    172.16.0.13
contrail-k8s-apiserver-6d79c8598d-szdzf     ok      ocp2    172.16.0.12
```

```
contrail-k8s-controller-96964f568-csk2k         ok      ocp1    172.16.0.11
contrail-k8s-controller-96964f568-dshn6         ok      ocp3    172.16.0.13
contrail-k8s-controller-96964f568-hfrpl         ok      ocp2    172.16.0.12
contrail-k8s-kubemanager-79b577ff86-6v8qt       ok      ocp3    172.16.0.13
contrail-k8s-kubemanager-79b577ff86-cbh5n       ok      ocp1    172.16.0.11
contrail-k8s-kubemanager-79b577ff86-vmckw       ok      ocp2    172.16.0.12


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


LOCAL BGPROUTER NEIGHBOR BGPROUTER      ENCODING        STATE           POD
ocp1            ocp2                    BGP             Established ok  contrail-control-0
ocp1            ocp3                    BGP             Established ok  contrail-control-0
ocp1            ocp1                    XMPP            Established ok  contrail-control-0
ocp1            ocp2                    XMPP            Established ok  contrail-control-0
ocp1            ocp3                    XMPP            Established ok  contrail-control-0
ocp1            ocp4                    XMPP            Established ok  contrail-control-0
ocp1            ocp5                    XMPP            Established ok  contrail-control-0
ocp2            ocp3                    BGP             Established ok  contrail-control-1
ocp2            ocp1                    BGP             Established ok  contrail-control-1
ocp2            ocp2                    XMPP            Established ok  contrail-control-1
ocp2            ocp3                    XMPP            Established ok  contrail-control-1
ocp2            ocp4                    XMPP            Established ok  contrail-control-1
ocp2            ocp5                    XMPP            Established ok  contrail-control-1
ocp3            ocp1                    BGP             Established ok  contrail-control-2
ocp3            ocp2                    BGP             Established ok  contrail-control-2
ocp3            ocp1                    XMPP            Established ok  contrail-control-2


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

Release Information

Table 7: Summary of Changes

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

4

CHAPTER

Manage

Manage Single Cluster CN2

SUMMARY

Learn how to perform life cycle management tasks in a single cluster installation.

IN THIS SECTION

Overview

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

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

Back Up the Contrail Etcd Database in Release 22.4

Use this example procedure in release 22.4 to back up the Contrail etcd database.

NOTE: The following steps refer to a Contrail controller node. A Contrail controller node is a worker node that is running a Contrail controller.

1. Install etcdctl on all Contrail controller nodes.
   a. Log in to one of the Contrail controller nodes.

For example:

```
ssh core@172.16.0.11
```

b. Download etcd. This example downloads to the **/tmp** directory.

```
ETCD_VER=v3.4.13
curl -L https://storage.googleapis.com/etcd/${ETCD_VER}/etcd-${ETCD_VER}-linux-
amd64.tar.gz -o /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz
```

c. Untar and move the etcd executable to a directory in your path (for example **/usr/local/bin**).

```
tar -xzvf /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz -C /tmp
sudo mv /tmp/etcd-${ETCD_VER}-linux-amd64/etcdctl /usr/local/bin
```

d. Check that you've installed etcd.

```
etcdctl version
etcdctl version: 3.4.13
API version: 3.4
```

e. Repeat on all the Contrail controller nodes.

2. Get a list of the contrail-etcd pods.

```
kubectl get pods -A | grep contrail-etcd
```

Take note of the contrail-etcd pod names, the IP addresses, and the nodes they're running on. You'll need this information in the next few steps.

3. Copy the etcd certificate and key files from the pods to the Contrail controller nodes.

We run kubectl on the Contrail controller nodes in this step. We assume you've set up kubeconfig on these nodes in its default location (**~/.kube/config**).

a. Pick a contrail-etcd pod (for example, contrail-etcd-0) and log in to the Contrail controller node that's hosting that pod.

b. Copy the certificate and key files from that contrail-etcd pod to the hosting Contrail controller node.

In this example, we're copying the certificates and key files from the contrail-etcd-0 pod to local files on this node.

```
kubectl exec --namespace contrail-system contrail-etcd-0 -c contrail-etcd -- cat /etc/
member-tls/ca.crt > ./ca.crt
kubectl exec --namespace contrail-system contrail-etcd-0 -c contrail-etcd -- cat /etc/
member-tls/tls.crt > ./tls.crt
kubectl exec --namespace contrail-system contrail-etcd-0 -c contrail-etcd -- cat /etc/
member-tls/tls.key > ./tls.key
```

This copies the certificate and key files from the contrail-etcd-0 pod to **ca.crt**, **tls.crt**, and **tls.key** in the current directory on this control plane node.

c. Repeat for each contrail-etcd pod.

4. Back up the etcd database on one of the Contrail controller nodes. You only need to back up the database on one node.

   a. Log back in to one of the control plane nodes.

   b. Back up the etcd database.

      This example saves the database to **/tmp/etcdbackup.db** on this Contrail controller node.

```
etcdctl snapshot save /tmp/etcdbackup.db --endpoints=<etcd-pod-ip>:<etcd-port>
 --cacert=ca.crt --cert=tls.crt --key=tls.key
```

      where *<etcd-pod-ip>* is the IP address of the pod on this node and the *<etcd-port>* is the port that etcd is listening on (by default, 12379).

5. Copy the database to a safe location.

Restore the Contrail Etcd Database in Release 22.4

Use this example procedure in release 22.4 to restore the Contrail etcd database from a snapshot on an Amazon EKS cluster.

NOTE: The following steps refer to a Contrail controller node. A Contrail controller node is a worker node that is running a Contrail controller.

1. Copy the snapshot you want to restore to all the Contrail controller nodes.

The steps below assume you've copied the snapshot to **/tmp/etcdbackup.db** on all the Contrail controller nodes.

2. Restore the snapshot.

   a. Log in to one of the Contrail controller nodes. In this example, we're logging in to the Contrail controller node that is hosting contrail-etcd-0.

   b. Restore the etcd database to the contrail-etcd-0 pod on this Contrail controller node.

      This creates a **contrail-etcd-0.etcd** directory on the node.

```
ETCDCTL_API=3 etcdctl snapshot restore /tmp/etcdBackup.db \
--name=contrail-etcd-0 \
--initial-cluster=contrail-etcd-0=https://<contrail-etcd-0-ip>:12380,\
contrail-etcd-1=https://<contrail-etcd-1-ip>:12380,\
contrail-etcd-2=https://<contrail-etcd-2-ip>:12380 \
--initial-advertise-peer-urls= https://<contrail-etcd-0-ip>:12380 \
--cacert=ca.crt --cert=tls.crt --key=tls.key
```

      where `--name=contrail-etcd-0` specifies that this command is restoring the database to contrail-etcd-0, `--initial-cluster=...` lists all the contrail-etcd members in the cluster, and `--initial-advertise-peer-urls=...` refers to the IP address and port number that the contrail-etcd-0 pod is listening on.

   c. Repeat for the other contrail-etcd pods on their respective Contrail controller nodes, substituting the `--name` and `--initial-advertise-peer-urls` values with the respective contrail-etcd pod name and IP address.

3. Stop the contrail-etcd pods.

   This sets the replicas to 0, which effectively stops the pods.

```
kubectl patch etcds.datastore.juniper.net contrail-etcd -n contrail-system --type=merge -p
'{"spec": {"common": {"replicas": 0}}}
```

4. Replace contrail-etcd data with the data from the snapshot.

   a. SSH into one of the Contrail controller nodes.

   b. Replace the data. Recall that the snapshot is stored in the **contrail-etcd-*<xxx>*.etcd** directory.

```
sudo rm -rf /var/lib/contrail-etcd/snapshots
sudo mv /var/lib/contrail-etcd/etcd/member /var/lib/contrail-etcd/etcd/member.bak
sudo mv contrail-etcd-<xxx>.etcd/member /var/lib/contrail-etcd/etcd/
```

where *contrail-etcd-xxx* is the name of the contrail-etcd pod on the Contrail controller node that you logged in to.

c. Repeat for the other Contrail controller nodes.

5. Start the contrail-etcd pods.

   This sets the replicas to 3, which effectively starts the pods.

```
kubectl patch etcds.datastore.juniper.net contrail-etcd -n contrail-system --type=merge -p
'{"spec": {"common": {"replicas": 3}}}
```

6. 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.

7. Restart the vrouters.

   Delete all the contrail-vrouter-nodes pods.

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

   These pods will automatically restart.

8. Check that all pods are in running state.

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

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

5
CHAPTER

Appendix

Configure Repository Credentials

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

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

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

   Enter your login credentials when prompted.

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

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

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

Run Preflight and Postflight Checks

SUMMARY

Learn about CN2 preflight and postflight checks.

IN THIS SECTION

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

Postflight checks allow you to verify that your CN2 installation is working properly. The checks test for status, pod-to-pod communication, API server reachability, and other basic functions. You run postflight checks after installing CN2. CN2 supports postflight checks starting in release 22.2.

Run Preflight and Postflight Checks in Release 22.4

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 CN2 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 37 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 CN2.

- To run the postflight checks:

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

You run postflight checks after you install CN2.

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

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.

Juniper CN2 Technology Previews (Tech Previews)

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

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

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

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

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