

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

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Cloud Native Contrail Networking Installation and Life Cycle Management Guide for Amazon EKS
Copyright © 2023 Juniper Networks, Inc. All rights reserved.

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

1

Introduction

Cloud-Native Contrail Networking Overview | 2

Terminology | 4

CN2 Components | 6

Deployment Models | 11

Single Cluster Deployment | 11

Multi-Cluster Deployment | 12

System Requirements | 15

2

Install

Overview | 17

Before You Install | 18

Install Single Cluster CN2 on Amazon EKS | 19

Install Single Cluster CN2 Using Amazon EKS Blueprints | 20

Install Single Cluster CN2 Using Helm Charts | 22

Install Single Cluster CN2 Using YAML Manifests | 23

Install Multi-Cluster CN2 on Amazon EKS | 25

Install Multi-Cluster CN2 | 25

Install Contrail Tools | 28

Install ContrailReadiness Controller | 28

Manifests | 29

Manifests in Release 23.2 | 30

Contrail Tools in Release 23.2 | 31

Contrail Analytics in Release 23.2 | 32

3

Monitor**Install Contrail Analytics | 34**

| Install Contrail Analytics and the CN2 Web UI | 34

Kubectl Contrailstatus | 37

4

Manage**Manage Single Cluster CN2 | 43**

| Overview | 43

| Run Preflight and Postflight Checks | 43

| Back Up the Contrail Etcd Database | 45

| Restore the Contrail Etcd Database | 47

| Upgrade CN2 | 49

| Uninstall CN2 | 51

Manage Multi-Cluster CN2 | 52

| Attach a Workload Cluster | 52

5

Appendix**Configure Repository Credentials | 63****Juniper CN2 Technology Previews (Tech Previews) | 64**

1

CHAPTER

Introduction

Cloud-Native Contrail Networking Overview | 2

Terminology | 4

CN2 Components | 6

Deployment Models | 11

System Requirements | 15

Cloud-Native Contrail Networking Overview

SUMMARY

Learn about Cloud-Native Contrail Networking (CN2).

IN THIS SECTION

- [Benefits of Cloud-Native Contrail Networking | 4](#)

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

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

In release 23.2, Cloud-Native Contrail Networking is supported on the following:

- [\(Upstream\) Kubernetes](#)
- [Red Hat Openshift](#)
- [Amazon EKS](#)
- [Rancher RKE2](#)

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

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

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

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

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

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

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

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

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

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

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

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

Benefits of Cloud-Native Contrail Networking

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

Terminology

Table 1: Terminology

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

Table 1: Terminology (*Continued*)

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

Table 1: Terminology (*Continued*)

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

CN2 Components

The CN2 architecture on Amazon EKS 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 Kubernetes control plane. The pods that perform the network configuration plane function reside on worker nodes that contain the Contrail controller.
- The network control plane represents CN2's full-featured SDN capability. It uses BGP to communicate with other Contrail controllers and XMPP to communicate with the distributed data plane components on all worker nodes. The pods that perform the network control plane function reside on worker nodes that contain the Contrail controller.
- The network data plane refers to the packet transmit and receive function on worker nodes. The pods that perform the network data plane function reside on all worker nodes.

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

Table 2: Main CN2 Components

Pod Name		Where	Description
Configuration Plane ¹	contrail-k8s-apiserver	Worker Node	<p>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.</p> <p>There is one contrail-k8s-apiserver pod per Contrail controller node.</p>
	contrail-k8s-controller	Worker Node	<p>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.</p> <p>There is one contrail-k8s-controller pod per Contrail controller node.</p>
	contrail-k8s-kubemanager	Worker Node	<p>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.</p> <p>In a single-cluster deployment, there is one contrail-k8s-kubemanager pod per Contrail controller node.</p> <p>In a multi-cluster deployment, there is additionally one contrail-k8s-kubemanager pod for every distributed workload cluster.</p>

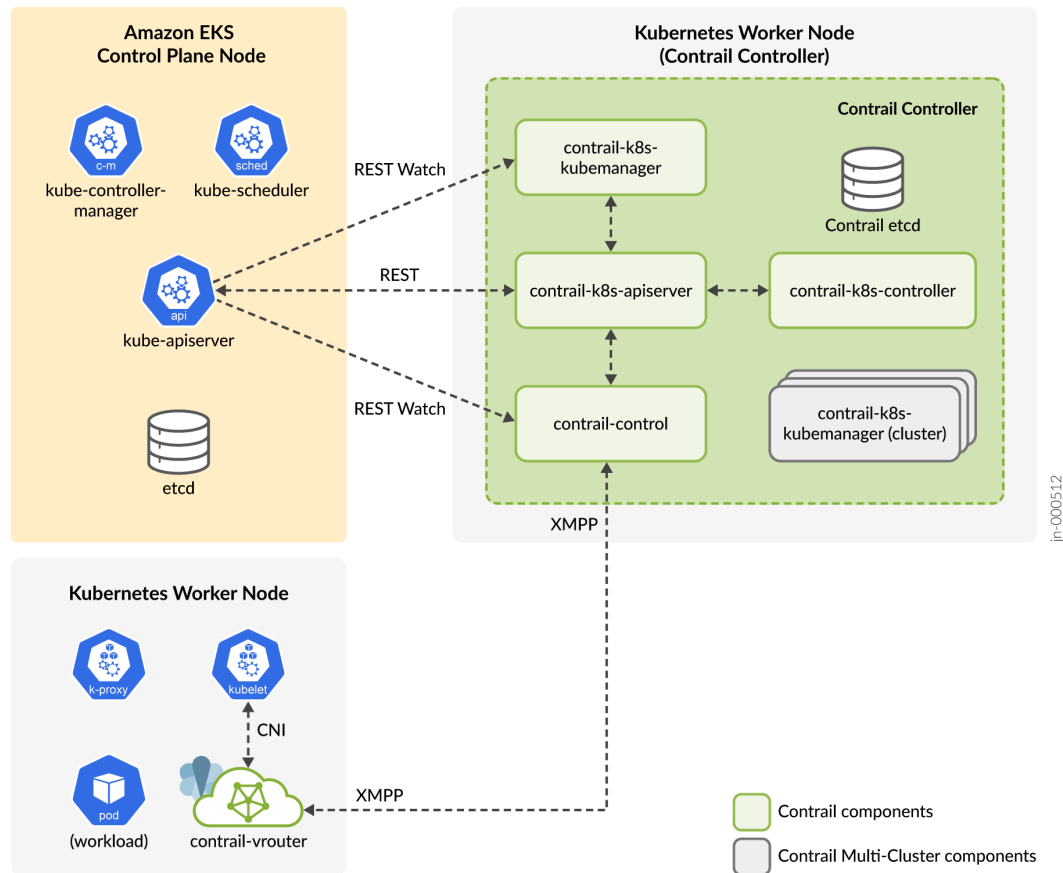
Table 2: Main CN2 Components *(Continued)*

Pod Name		Where	Description
Control Plane ¹	contrail-control	Worker Node	<p>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. It then communicates these changes with its BGP peers and vRouter agents (over XMPP) as appropriate.</p> <p>There is one contrail-control pod per Contrail controller node.</p>
	contrail-vrouter-nodes	Worker Node	<p>This pod contains the vRouter agent and the vRouter.</p> <p>The vRouter agent acts on behalf of the local vRouter when interacting with the Contrail controller. There is one agent per node. The agent establishes XMPP sessions with two Contrail controllers to perform the following functions:</p> <ul style="list-style-type: none"> • translates configuration from the control plane into objects that the vRouter understands • interfaces with the control plane for the management of routes • collects and exports statistics from the data plane <p>The vRouter provides the packet send and receive function for the co-located pods and workloads. It provides the CNI plug-in functionality.</p>
<p>¹The components that make up the network configuration plane and the network control plane are collectively called the Contrail controller.</p>			

Figure 1 on page 9 shows these components in the context of an Amazon EKS cluster.

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

Figure 1: CN2 Components



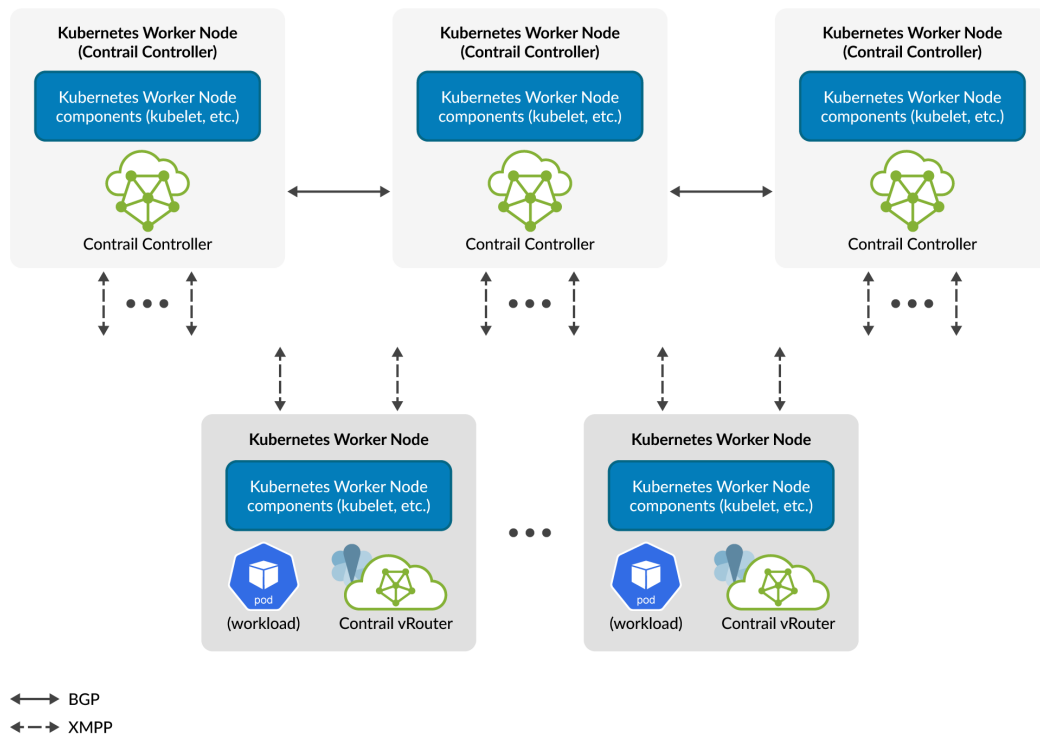
The cluster operates in standard Kubernetes fashion, with the kube-apiserver being the REST API entry point for the cluster. The kube-apiserver directs all networking requests to the contrail-k8s-apiserver, which translates them into REST API calls to the respective CN2 objects. In some cases, these calls may result in the Contrail controller sending XMPP messages to the vRouter agent on other worker nodes or sending BGP messages (not shown) to other Contrail controller nodes or external routers. The Contrail controller sends these XMPP and BGP messages outside of regular Kubernetes cluster communications.

The contrail-k8s-kubemanager (cluster) components are only present in multi-cluster deployments. For more information on the different types of deployments, see ["Deployment Models" on page 11](#).

The Contrail controller stores all CN2 cluster data in the Contrail etcd, separate from the main Amazon EKS etcd store.

Figure 2 on page 10 shows a cluster with multiple Contrail controllers. As before, the Kubernetes components communicate with each other using REST (not shown) and the Contrail controllers communicate with the vRouters using XMPP. For redundancy, each vRouter establishes XMPP communications with two Contrail controllers. Additionally, the Contrail controllers exchange routes with each other using iBGP. By doing so, if any Contrail controller goes down, the other Contrail controllers have enough information to keep the network control plane running.

Figure 2: Multiple Contrail Controllers



Deployment Models

SUMMARY

Learn about single cluster and multi-cluster CN2.

IN THIS SECTION

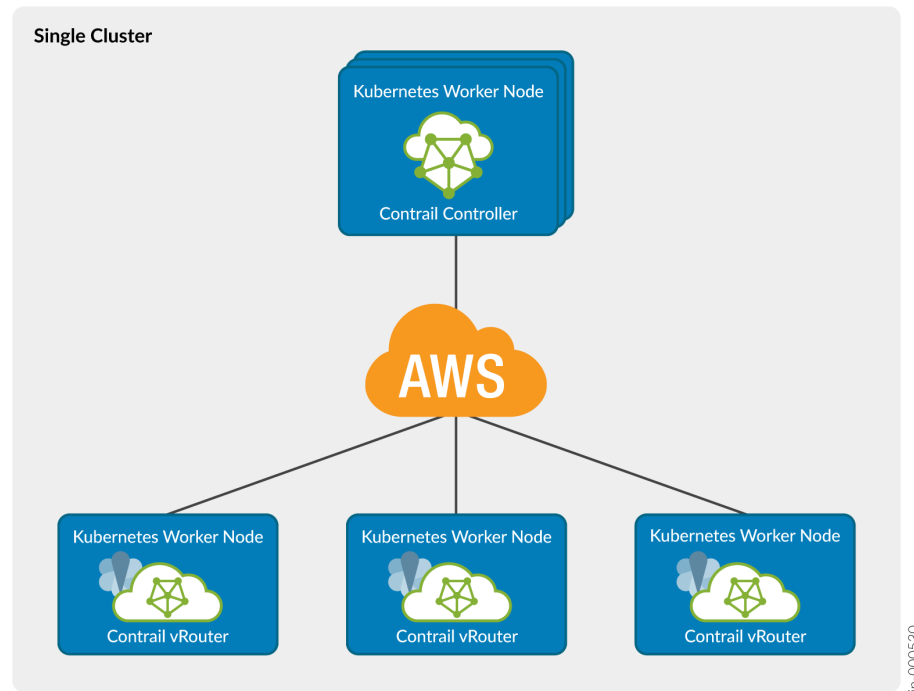
- [Single Cluster Deployment | 11](#)
- [Multi-Cluster Deployment | 12](#)

Single Cluster Deployment

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

In a single-cluster deployment on Amazon EKS ([Figure 3 on page 12](#)), 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.

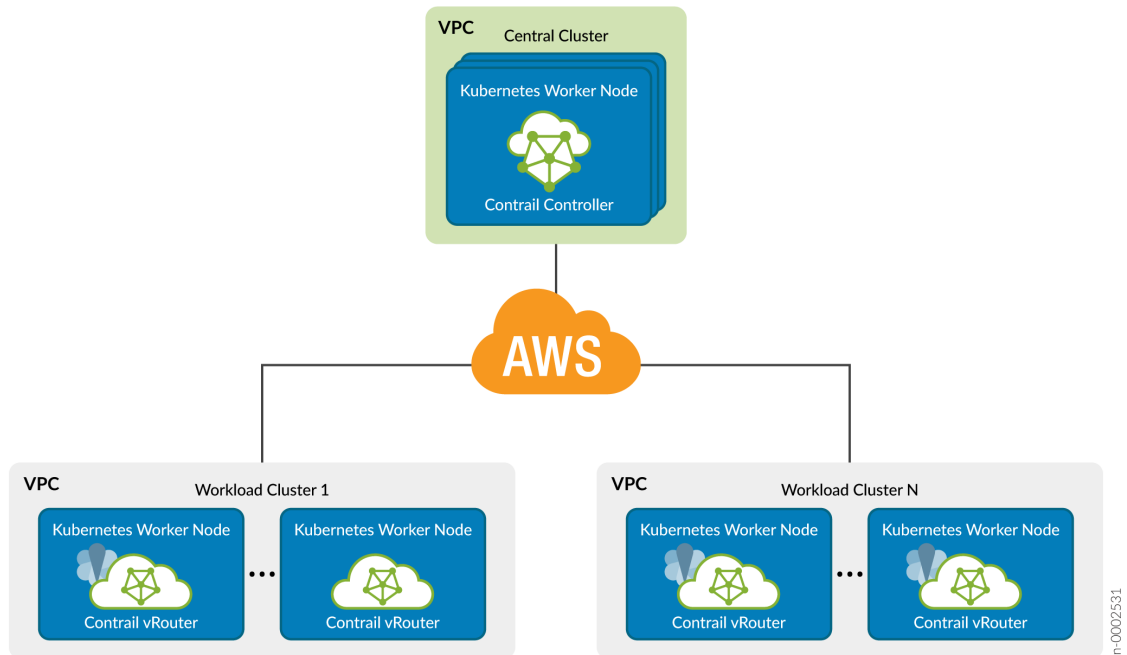
Figure 3: Single Cluster Deployment



Multi-Cluster Deployment

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

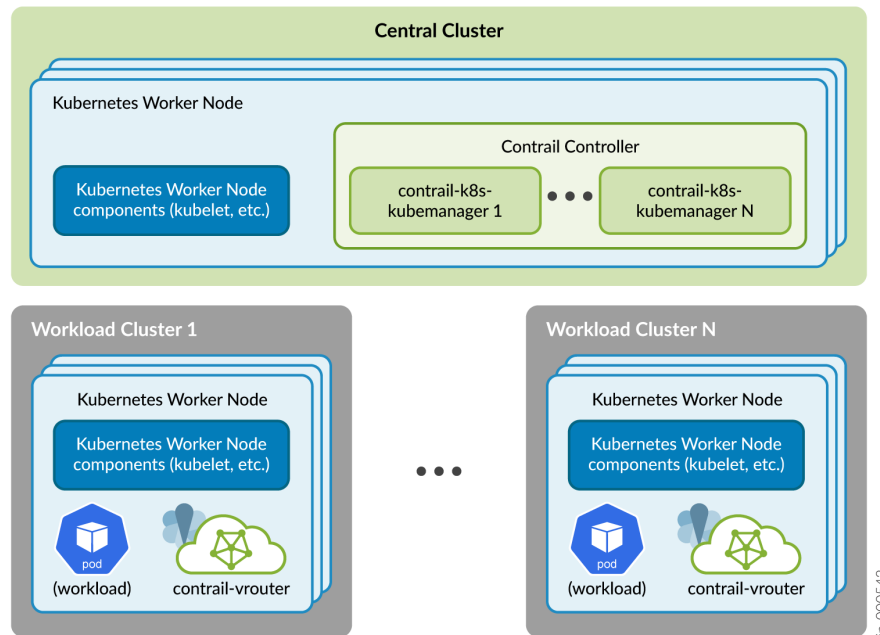
Figure 4: Multi-Cluster Deployment



Centralizing the networking in this way makes the networking easier to configure and manage and easier to apply consistent network policy and security.

[Figure 4 on page 13](#) provides more detail on this setup. The Contrail controller sits in the central cluster and contains a kubemanager for each distributed workload cluster that it serves. The distributed workload cluster runs the workloads and contains the Contrail vRouter.

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 Amazon EKS Installation with CN2

Machine	Instance Type	Notes
Control Plane Nodes	Not applicable	Outside of user control. The Amazon EKS control plane scales automatically 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	Depends on user workload	For worker nodes running user workloads and the Contrail data plane (no Contrail controller).

2

CHAPTER

Install

[Overview](#) | 17

[Before You Install](#) | 18

[Install Single Cluster CN2 on Amazon EKS](#) | 19

[Install Multi-Cluster CN2 on Amazon EKS](#) | 25

[Install Contrail Tools](#) | 28

[Manifests](#) | 29

Overview

IN THIS SECTION

- [Benefits of Amazon EKS with CN2 | 17](#)

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.

We provide multiple ways to help you to install CN2 in an Amazon EKS cluster. You can use the provided Amazon EKS blueprints for Terraform, the provided Helm charts, or the provided YAML manifests.

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

You can install CN2 on Amazon EKS using Amazon EKS blueprints for Terraform, Helm charts, or YAML manifests.

If you're looking for a quick proof-of-concept installation, then use the Amazon EKS blueprints. Our blueprint creates an Amazon EKS cluster and then installs CN2 on that cluster using the YAML manifests that we prepopulate in the blueprint repository.

If you already have an Amazon EKS cluster, then use the Helm charts or use the YAML manifests directly. Helm charts are slightly easier to use, but YAML manifests provide you the most flexibility.

1. Set up an AWS account and a Juniper Networks account.

You'll need the AWS account to create an Amazon EKS cluster and you'll need the Juniper Networks account to download CN2 manifests from the Juniper Networks download site (<https://support.juniper.net/support/downloads/?p=contrail-networking>) and to access the container repository at <https://enterprise-hub.juniper.net>.

2. Install the AWS CLI. See <https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>.
3. Configure the AWS CLI with your AWS access key and secret. See <https://awscli.amazonaws.com/v2/documentation/api/latest/reference/configure/index.html>.
4. Install eksctl. See <https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html>.
5. Install kubectl. See <https://kubernetes.io/docs/tasks/tools/>.
6. Install Helm. See <https://helm.sh/docs/intro/install/>.

7. If you're planning on using Amazon EKS blueprints, then install Terraform:

- a. Install Terraform. See <https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli>.
- b. 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.

8. If you're planning on using the CN2 YAML manifests directly, then download the manifests:

- a. Download the Contrail Networking manifests. See ["Manifests" on page 29](#).
- b. Configure your repository login credentials in the downloaded manifests.

Add your repository login credentials to the **amazon-eks** and **contrail-tools** manifests. See ["Configure Repository Credentials" on page 63](#) for one way to do this.

9. Optionally, install contrailstatus. Contrailstatus is a kubectl plug-in you can use to query CN2 components and resources. See ["Manifests" on page 29](#).

Change permissions on the **kubectl-contrailstatus** executable and copy it somewhere in your path, for example **/usr/local/bin**.

```
chmod +x kubectl-contrailstatus
```

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

Install Single Cluster CN2 on Amazon EKS

SUMMARY

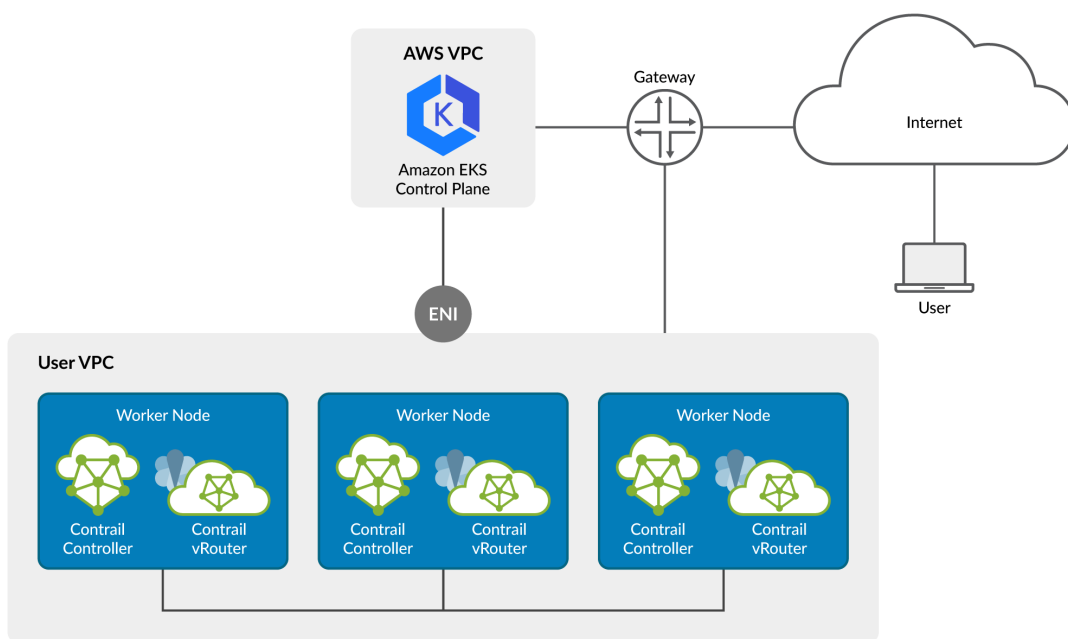
See examples on how to install single cluster CN2 on Amazon EKS.

IN THIS SECTION

- [Install Single Cluster CN2 Using Amazon EKS Blueprints | 20](#)
- [Install Single Cluster CN2 Using Helm Charts | 22](#)
- [Install Single Cluster CN2 Using YAML Manifests | 23](#)

In a single cluster deployment, CN2 is the networking platform and CNI plug-in for that cluster. [Figure 6 on page 20](#) shows an Amazon EKS cluster with three worker nodes running the Contrail controller. The Amazon EKS control plane communicates with worker nodes in the user VPC over an Elastic Network Interface (ENI). In a typical deployment, there would be additional worker nodes that run the user workloads.

Figure 6: CN2 on Amazon EKS



The procedures in this section show basic examples of how you can use the provided Amazon EKS blueprints, Helm charts, and YAML manifests to install CN2 on an Amazon EKS cluster. We cover both installing CN2 in a brand new cluster and in an existing cluster.

You're not limited to the deployment described in these sections nor are you limited to using the provided files and manifests. CN2 supports a wide range of deployments that are too numerous to cover in detail. Use the provided examples as a starting point to roll your own manifest tailored to your specific situation.

Install Single Cluster CN2 Using Amazon EKS Blueprints

Use this procedure to install CN2 using Amazon EKS blueprints for Terraform.

The blueprint that we provide performs the following:

- 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 -b awseks-23.1 --single-branch
```

NOTE: There is no release 23.2 branch. Use the release 23.1 branch instead.

2. Add your enterprise-hub.juniper.net access credentials to **terraform-aws-eks-blueprints/examples/eks-cluster-with-cn2/variables.tf** for the `container_pull_secret` variable .

The credentials that you add must be base64-encoded. See ["Configure Repository Credentials" on page 63](#) for an example of how to obtain and encode your credentials.

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 apply`. 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:

```
kubectl get nodes
```

List all the pods:

```
kubectl get pods -A
```

9. (Optional) Run postflight checks. See ["Run Preflight and Postflight Checks" on page 43](#).

10. If you run into problems, clean up the cluster and try the installation again.

To clean up the cluster, destroy the Kubernetes addons, the Amazon EKS cluster, and the VPC. You must run these terraform commands in the **examples/eks-cluster-with-cn2** directory.

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

```
terraform destroy -target="module.eks_blueprints_kubernetes_addons" -auto-approve
terraform destroy -target="module.eks_blueprints" -auto-approve
terraform destroy -target="module.vpc" -auto-approve
```

Then destroy any remaining resources:

```
terraform destroy -auto-approve
```

Install Single Cluster CN2 Using Helm Charts

Use this procedure to install CN2 on an existing Amazon EKS cluster using Helm charts. In this example, the existing Amazon EKS cluster is running the VPC CNI.

1. Add the Juniper Networks CN2 Helm repository.

```
helm repo add cn2 https://juniper.github.io/cn2-helm/
```

2. Install CN2.

```
helm install cn2eks cn2/cn2-eks --set imagePullSecret="<base64-encoded-credential>"
```

See ["Configure Repository Credentials" on page 63](#) for one way to get your credentials.

3. Use standard kubectl commands to check on the installation.

```
kubectl get nodes
```

Check that the nodes are up. If the nodes are not up, wait a few minutes and check again.

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

Check that the pods have a STATUS of Running. If not, wait a few minutes for the pods to come up.

4. (Optional) Run postflight checks. See ["Run Preflight and Postflight Checks" on page 43](#).

Install Single Cluster CN2 Using YAML Manifests

Use this procedure to install CN2 using YAML manifests.

We use eksctl to create a cluster in this example, but you can use any other method as long as you remember to remove the CN2.

The manifests that you will use in this example procedure are **amazon-eks/single-cluster/single_cluster_deployer_example.yaml** and **amazon-eks/single-cluster/cert-manager.yaml**. The procedure assumes that you've placed these manifests into a **manifests** directory.

1. Create an EKS cluster without a node group.

```
eksctl create cluster --name mycluster --without-nodegroup
```

Take note of the service IP address subnet. You'll need this in a later step. By default, Amazon EKS assigns service IP addresses from either the 10.100.0.0/16 or the 172.20.0.0/16 CIDR blocks.

```
eksctl get cluster --name mycluster -o json | jq .[0].KubernetesNetworkConfig
```

2. Configure the service IP address subnet for the Contrail kubemanager. This subnet must match the service IP address subnet of the cluster.

Edit the **single_cluster_deployer_example.yaml** manifest and look for the serviceV4Subnet configuration in the Kubemanager section.

```
serviceV4Subnet: 172.20.0.0/16
```

Change the subnet as necessary to match the service IP address subnet of the cluster.

3. If desired, specify the three nodes where you want to run the Contrail controller.

By default, the supplied manifest contains tolerations that allow the Contrail controller to tolerate any taint. This means that the Contrail controller will install on any node. Use node selectors (or node affinity) to force the Contrail controller to install on the nodes that you want. Then taint those nodes to prevent other pods from being scheduled there. Repeat for the other two nodes.

4. Apply the cert-manager manifest. The cert-manager provides encryption for all CN2 management and control plane connections.

```
kubectl apply -f manifests/cert-manager.yaml
```

5. Apply the Contrail deployer manifest.

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

6. Attach managed or self-managed worker nodes running Amazon EKS-optimized AMI to the cluster. Ensure you pick an AMI that is running a kernel that is supported by CN2.

```
eksctl create nodegroup --cluster mycluster --node-type m5.xlarge --node-ami-family  
AmazonLinux2 --max-pods-per-node 100 --node-private-networking
```

7. (Optional) Install Contrail tools and run preflight checks. See ["Run Preflight and Postflight Checks"](#) on [page 43](#).

Correct any errors before proceeding.

8. Use standard kubectl commands to check on the deployment.

```
kubectl get nodes
```

Check that the nodes are up. If the nodes are not up, wait a few minutes and check again.

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

Check that the pods have a STATUS of Running. If not, wait a few minutes for the pods to come up.

9. (Optional) Run postflight checks. See ["Run Preflight and Postflight Checks" on page 43](#).

Install Multi-Cluster CN2 on Amazon EKS

SUMMARY

See examples on how to install multi-cluster CN2 on Amazon EKS.

IN THIS SECTION

- [Install Multi-Cluster CN2 | 25](#)

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

To install CN2 in a multi-cluster deployment, you first create the central cluster and then you attach the distributed workload clusters to the central cluster one by one.

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

Install Multi-Cluster CN2

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

We use eksctl and YAML manifests to set up the clusters in this example.

The CN2 manifest that we use in this example procedure is **amazon-eks/multi-cluster/central_cluster_deployer_example.yaml** and **amazon-eks/multi-cluster/central_cluster_cert-manager.yaml**. The procedure assumes that you've placed these manifests into a **manifests** directory.

1. Create the central cluster with no worker nodes.

- a. Create a yaml file that describes the cluster. We'll name this file **eksctl-central-cluster.yaml**.

For example:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: cn2-m-central
  region: us-west-2
kubernetesNetworkConfig:
  serviceIPv4CIDR: 10.11.0.0/16
vpc:
  cidr: 192.168.0.0/16
```

Populate the file with your desired values:

- **name** - the name you want to call the cluster
- **region** - the AWS region of your cluster
- **serviceIPv4CIDR** - the IP address subnet you want to assign to Kubernetes services
- **cidr** - the IP address subnet you want to assign to your VPC

- b. Apply this YAML to create the central cluster (without a node group).

```
eksctl create cluster --without-nodgroup --config-file eksctl-central-cluster.yaml
```

- c. Store the central cluster's region and VPC CIDR into variables for later use.

For example:

```
export CENTRAL_REGION=us-west-2
```

```
export CENTRAL_VPC_CIDR=192.168.0.0/16
```

- d. Store the central cluster's VPC ID into a variable for later use.

```
export CENTRAL_VPC_ID=$(eksctl get cluster --name cn2-m-central -o json --region
$CENTRAL_REGION | jq -r .[0].ResourcesVpcConfig.VpcId) && echo $CENTRAL_VPC_ID
```

- e. Create a security group to accept traffic from future workload cluster VPCs.

```
CENTRAL_SG_ID=$(aws ec2 create-security-group --region $CENTRAL_REGION --vpc-id
$CENTRAL_VPC_ID --group-name workload-to-central --description "Allow workload to central"
| jq -r .GroupId) && echo $CENTRAL_SG_ID
```

This security group is just a placeholder for now. When we start adding distributed workload clusters, we'll add rules to this security group to govern traffic from those distributed workload clusters.

2. Install CN2 on the central cluster.

- a. Apply the central cluster manifest.

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

- b. Apply the central cluster cert-manager manifest.

```
kubectl apply -f manifests/central_cluster_cert-manager.yaml
```

- c. Add worker nodes to the cluster and specify the security group.

```
eksctl create nodegroup --region $CENTRAL_REGION --cluster cn2-m-central --node-type
m5.large --node-ami-family AmazonLinux2 --max-pods-per-node 100 --node-private-networking
--node-security-groups $CENTRAL_SG_ID
```

- d. 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 52](#) to create and attach a distributed workload cluster to this central cluster.

4. Repeat step 3 for every distributed workload cluster you want to create and attach.
5. (Optional) Run postflight checks. See ["Run Preflight and Postflight Checks" on page 43](#).

NOTE: Run postflight checks from the central cluster only.

Install Contrail Tools

SUMMARY

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

IN THIS SECTION

- [Install ContrailReadiness Controller | 28](#)

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

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

- ["Run Preflight and Postflight Checks" on page 43](#)
- ["Run Preflight and Postflight Checks" on page 43](#)
- ["Uninstall CN2" on page 51](#)

Install ContrailReadiness Controller

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

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

1. Locate the **contrail-tools/contrail-readiness** directory from the downloaded CN2 Tools package.
2. If you haven't already done so, ensure you've populated the tools manifests with your repository login credentials. See ["Configure Repository Credentials" on page 63](#) for one way to do this.

3. Apply the ContrailReadiness custom resource definitions.

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

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

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

where `<path_to_deployer_manifest>` is the full path to the deployer manifest that you want to apply or have applied.

5. Patch the ConfigMap with the registry information.

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

6. Create the ContrailReadiness controller.

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

Check that the controller has come up.

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

Manifests

SUMMARY

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

IN THIS SECTION

- [Manifests in Release 23.2 | 30](#)
- [Contrail Tools in Release 23.2 | 31](#)
- [Contrail Analytics in Release 23.2 | 32](#)

Manifests in Release 23.2

The Manifests for Amazon EKS 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/23.2>).

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

If you're downloading from the Juniper Networks software download site, you'll need an account to download. If you don't have an account, contact your Juniper Networks sales representative to have one created for you. The package you download is called **contrail-manifests-eks-23.2.0.157.tgz**.

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

Table 4: Single Cluster Manifests for Amazon EKS for Release 23.2

Manifests	Description
amazon-eks/single_cluster/cert-manager.yaml	Contrail cert-manager manifests for encrypting Contrail management and control plane communications.
amazon-eks/single_cluster/single_cluster_deployer_example.yaml	Contains the manifests to install Contrail in a single cluster.

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

Table 5: Multi-Cluster Manifests for Amazon EKS for Release 23.2

Manifests	Description
amazon-eks/multi-cluster/central_cluster_cert-manager.yaml	Contrail cert-manager manifests for encrypting Contrail management and control plane communications on the central cluster.
amazon-eks/multi-cluster/central_cluster_deployer_example.yaml	Contrail deployer and necessary resources for the central cluster in a multi-cluster setup.

Table 5: Multi-Cluster Manifests for Amazon EKS for Release 23.2 (*Continued*)

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

Contrail Tools in Release 23.2

IN THIS SECTION

- [Contrail Tools | 31](#)

Contrail Tools

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

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

The release 23.2 package is called **contrail-tools-23.2.0.157.tgz**.

The following table lists tools that we provide.

Table 6: Tools Manifests for Release 23.2

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/contrail-readiness-uninstall.yaml	ContrailReadiness uninstall custom resource
contrail-tools/contrail-readiness/crds	ContrailReadiness custom resource definitions for the supported tools
contrail-tools/kubectl-contrailstatus- <release>.tar	The kubectl contrailstatus plug-in
contrail-tools/cn2_debug_infra- <release>.tar	The CN2 debug utility
contrail-tools/uninstall.tar.gz	Deprecated

Contrail Analytics in Release 23.2

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 23.2 package is called **contrail-analytics-23.2.0.157.tgz**.

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

3

CHAPTER

Monitor

[Install Contrail Analytics](#) | 34

[Kubectl Contrailstatus](#) | 37

Install Contrail Analytics

SUMMARY

Learn how to install Contrail Analytics and the CN2 Web UI.

IN THIS SECTION

- [Install Contrail Analytics and the CN2 Web UI | 34](#)

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. Also included is the CN2 Web UI, which allows you to monitor and configure CN2 components.

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

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 and the CN2 Web UI

Use this procedure to install Contail Analytics and the CN2 Web UI 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 --create-namespace
```

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

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

Alternatively, you can specify an Internet-facing AWS load balancer service by adding `--set enableLoadBalancer=true`. 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 or `HOSTNAME`.

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 64 or contact [Juniper Support](#).

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

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

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

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

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

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

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

Alternatively, you can specify an Internet-facing AWS load balancer service by adding `--set enableLoadBalancer=true`. 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 or `HOSTNAME`.

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

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

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

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

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

To access the CN2 Web UI, point your browser to `https://<node-IP-address>:30443`. You can log in to your cluster using your kubeconfig.

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

For additional information, refer to the ["Juniper CN2 Technology Previews \(Tech Previews\)" on page 64](#) or contact [Juniper Support](#).

6. To uninstall Contrail Analytics:

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

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

7. To upgrade Contrail Analytics:

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

or (for upgrading HA)

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


Kubectl Contrailstatus

IN THIS SECTION

- [Syntax | 37](#)
- [Description | 37](#)
- [Options | 38](#)
- [Additional Information | 38](#)
- [Output Fields | 39](#)
- [Sample Output | 40](#)
- [Release Information | 41](#)

Syntax

```
kubectl contrailstatus deployment --plane { config | control | data [--wide] }  
kubectl contrailstatus resource { bgprouter [BGP | XMPP] | globalssystemconfig | 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

kubect! contrailstatus deployment --plane config	Displays the status of the Configuration plane components: <ul style="list-style-type: none"> • contrail-k8s-apiserver • contrail-k8s-controller • contrail-k8s-kubemanager
kubect! contrailstatus deployment --plane control	Displays the status of the Control plane components: <ul style="list-style-type: none"> • contrail-control
kubect! contrailstatus deployment --plane data	Displays the status of the Data plane components: <ul style="list-style-type: none"> • contrail-vrouter-masters • contrail-vrouter-nodes
kubect! contrailstatus resource bgprouter	Displays the status' of the various BGP and XMPP neighbor relationships.
kubect! contrailstatus resource globalsystemconfig	Displays the status of the GlobalSystemConfig.
kubect! contrailstatus resource routinginstance	Displays the status' of the various RoutingInstances in CN2.
kubect! contrailstatus resource virtualnetwork	Displays the status' of the various VirtualNetworks in CN2.
kubect! 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.
kubect! contrailstatus configdump	Lists the resources and their quantities.
kubect! contrailstatus --all	Displays the status' of the Configuration/Control/Data planes and the BGP and XMPP relationships.
kubect! contrailstatus version	Displays the versions of the various container images.

Additional Information

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

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

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

Output Fields

Table 7 on page 39 lists some of the output fields for the `kubectl contrailstatus` command.

Table 7: kubectl contrailstatus Output Fields

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

Sample Output

kubectrl contrail-status --all

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

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

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

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

ocp3	ocp1	BGP	Established ok	contrail-control-2
ocp3	ocp2	BGP	Established ok	contrail-control-2
ocp3	ocp1	XMPP	Established ok	contrail-control-2
NAME(DATA)	STATUS	NODE	IP	MESSAGE
contrail-vrouter-masters-dspzb	ok	ocp3	172.16.0.13	
contrail-vrouter-masters-ks249	ok	ocp2	172.16.0.12	
contrail-vrouter-masters-tn6jz	ok	ocp1	172.16.0.11	
contrail-vrouter-nodes-mjwt2	ok	ocp4	172.16.0.14	
contrail-vrouter-nodes-rp5np	ok	ocp5	172.16.0.15	

Release Information

Table 8: Summary of Changes

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

4

CHAPTER

Manage

[Manage Single Cluster CN2](#) | 43

[Manage Multi-Cluster CN2](#) | 52

Manage Single Cluster CN2

SUMMARY

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

IN THIS SECTION

- [Overview | 43](#)
- [Run Preflight and Postflight Checks | 43](#)
- [Back Up the Contrail Etcd Database | 45](#)
- [Restore the Contrail Etcd Database | 47](#)
- [Upgrade CN2 | 49](#)
- [Uninstall CN2 | 51](#)

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 other Kubernetes components to provide the networking infrastructure.

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

In addition to standard Kubernetes management tools, you can use tools and procedures that are specific to CN2. This section covers these tools and procedures. ["Install Contrail Tools" on page 28](#) ["Install ContrailReadiness Controller" on page 28](#)

Run Preflight and Postflight Checks

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

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

typically run preflight checks prior to installing CN2, but you can run these checks after installing CN2 as well.

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

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

1. Locate the **contrail-tools/contrail-readiness** directory from the downloaded CN2 Tools package.
2. If you haven't already done so, ensure you've populated the manifests with your repository login credentials. See ["Configure Repository Credentials" on page 63](#) for one way to do this.
3. To run the preflight checks:

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

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

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

4. To run the postflight checks:

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

You run postflight checks after you install CN2.

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

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

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

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

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

```
kubectl get contrailreadiness task 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.

Back Up the Contrail Etcd Database

Use this example procedure 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.
 - 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 -xzf /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 Contrail controller 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

Use this example procedure 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
```

Upgrade CN2

Use this procedure to upgrade CN2.

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

The CN2 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, CN2 data plane upgrades are not hitless. If desired, migrate traffic off of the node being upgraded prior to performing the upgrade.

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

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

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

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

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

4. Upgrade CN2.

```
kubectl apply -f manifests/cert-manager.yaml
```

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

The pods in each Deployment and Stateful set will upgrade one at a time. The vRouter DaemonSet will go down and come back up.

5. Use standard kubectl commands to check on the upgrade.

Check the status of the nodes.

```
kubectl get nodes
```

Check the status of the pods.

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

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

Uninstall CN2

Use this procedure to uninstall CN2.

This tool removes the following:

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

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

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

1. Locate the **contrail-tools/contrail-readiness** directory from the downloaded CN2 Tools package.
2. If you haven't already done so, ensure you've populated the manifests with your repository login credentials. See ["Configure Repository Credentials" on page 63](#) for one way to do this.
3. If you've installed Contrail Analytics, uninstall it now. The uninstall script does not uninstall resources in namespaces other than those listed above. To uninstall Contrail Analytics, see the *Install Contrail Analytics and the CN2 Web UI* section.
4. Delete any other resources and namespaces (for example, overlay networks) that you created after you installed CN2.
5. Uninstall CN2.

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

6. Query the uninstall results.

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

7. Finally, delete the contrail-readiness namespace.

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

Manage Multi-Cluster CN2

SUMMARY

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

IN THIS SECTION

- [Attach a Workload Cluster | 52](#)

This section covers tasks that are specific to a multi-cluster installation. If you want to perform management tasks in a specific cluster within the multi-cluster installation, then see "[Manage Single Cluster CN2](#)" on page 43.

Attach a Workload Cluster

Use this procedure to create and attach a distributed workload cluster to an existing central cluster.

The general procedure is:

1. Create the workload cluster (without a node group).
2. Enable communications between the workload cluster and the central cluster:
 - Allow connectivity between the workload cluster VPC and the central cluster VPC. You can accomplish this in a number of ways. The method that we'll use is VPC peering.
 - Create routes for traffic to flow between the two clusters.
 - Configure secrets to authenticate Kubernetes control plane traffic between the two clusters.

3. On the central cluster, create the kubemanager that manages the new distributed workload cluster.
4. Apply the CN2 distributed workload cluster manifest and add a node group.

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

NOTE: Before starting, make sure you've created the central cluster. See ["Install Multi-Cluster CN2" on page 25](#).

1. Store relevant central cluster information into variables that we'll use later.
If you're coming directly from installing the central cluster in ["Install Multi-Cluster CN2" on page 25](#), then skip this step because you've already set the variables.
Otherwise:

```
export CENTRAL_REGION=us-west-2
```

```
export CENTRAL_VPC_CIDR=192.168.0.0/16
```

```
export CENTRAL_VPC_ID=$(eksctl get cluster --name cn2-m-central -o json --region  
$CENTRAL_REGION | jq -r '[0].ResourcesVpcConfig.VpcId)
```

2. Create the distributed workload cluster. Do not add node groups to the cluster yet.
 - a. Create a yaml file that describes the cluster. We'll name this file **eksctl-workload-cluster.yaml**.
For example:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: cn2-m-workload
  region: us-east-2
kubernetesNetworkConfig:
  serviceIPv4CIDR: 172.30.0.0/16
```

```
vpc:
  cidr: 10.120.0.0/16
```

Populate the file with your desired values:

- name - the name you want to call the cluster
- region - the AWS region of your cluster
- serviceIPv4CIDR - the IP address subnet you want to assign to Kubernetes services
- cidr - the IP address subnet you want to assign to your VPC

Make sure the service IP and VPC CIDRs in the workload cluster differ from those in the central cluster.

- b. Apply this yaml to create the distributed workload cluster (without node groups).

```
eksctl create cluster --without-nodegroup --config-file eksctl-workload-cluster.yaml
```

- c. Store the workload cluster's region and VPC CIDR into variables for later use.

For example:

```
export WORKLOAD_REGION=us-east-2
```

```
export WORKLOAD_VPC_CIDR=10.120.0.0/16
```

3. Show the contexts of the clusters.

```
kubectl config get-contexts
```

CURRENT	NAME	CLUSTER	AUTHINFO	NAMESPACE
*	<central-cluster-context>	cn2-m-central.us-west-2.eksctl.io	<central-cluster-auth>	
	<workload-cluster-context>	cn2-m-workload.us-east-2.eksctl.io	<workload-cluster-auth>	

When we issue kubectl commands later, we'll need to direct the commands to the desired cluster. We'll use <central-cluster-context> and <workload-cluster-context> for that purpose.

4. Configure VPC peering to allow the central cluster VPC and the distributed workload cluster VPC to communicate with each other.

- a. Store the workload VPC ID into a variable for later use.

```
WORKLOAD_VPC_ID=$(eksctl get cluster --name cn2-m-workload -o json --region
$WORKLOAD_REGION | jq -r .[0].ResourcesVpcConfig.VpcId) && echo $WORKLOAD_VPC_ID
```

- b. Create the peering request and store into a variable for later use.

```
PEER_ID=$(aws ec2 create-vpc-peering-connection --vpc-id="$CENTRAL_VPC_ID" --peer-vpc-
id="$WORKLOAD_VPC_ID" --peer-region="$WORKLOAD_REGION" --query
VpcPeeringConnection.VpcPeeringConnectionId --output text) && echo $PEER_ID
```

- c. Accept the peering request.

```
aws ec2 accept-vpc-peering-connection --region $WORKLOAD_REGION --vpc-peering-connection-
id $PEER_ID
```

5. Add routes to each VPC so that the central and workload cluster VPCs can route to each other.

- a. Add a route to the central cluster VPC to allow it to reach the workload cluster VPC.

```
for rtb in $(aws ec2 describe-route-tables --region $CENTRAL_REGION --filters Name=vpc-
id,Values="$CENTRAL_VPC_ID" --query 'RouteTables[*].RouteTableId' | jq -r .[])
do
    aws ec2 create-route --region $CENTRAL_REGION --route-table-id "$rtb" --destination-
cidr-block "$WORKLOAD_VPC_CIDR" --vpc-peering-connection-id "$PEER_ID"
done
```

- b. Add a route to the workload cluster VPC to allow it to reach the central cluster VPC.

```
for rtb in $(aws ec2 describe-route-tables --region $WORKLOAD_REGION --filters Name=vpc-
id,Values="$WORKLOAD_VPC_ID" --query 'RouteTables[*].RouteTableId' | jq -r .[])
do
    aws ec2 create-route --region $DIST_REGION --route-table-id "$rtb" --destination-
cidr-block "$CENTRAL_VPC_CIDR" --vpc-peering-connection-id "$PEER_ID"
done
```

6. Configure the appropriate central cluster VPC security group to accept traffic from this workload cluster VPC.

If you're coming to this procedure directly from ["Install Multi-Cluster CN2" on page 25](#), then go straight to step 6.c because you've already set the `CENTRAL_SG_ID` variable.

- a. Find the security group ID for the security group by looking up the EC2 instance ID of the central cluster.

```
aws ec2 describe-instances --instance-ids <ec2-instance-id> | grep -A10 SecurityGroups
```

Look in the output for the security group you created in ["Install Multi-Cluster CN2" on page 25](#). If you're following this example, the security group is called `workload-to-central`.

- b. Store the `GroupId` corresponding to `workload-to-central` into a variable.

```
export CENTRAL_SG_ID=<GroupId>
```

- c. Add a rule to allow incoming traffic from the workload cluster VPC.

```
aws ec2 authorize-security-group-ingress --region $CENTRAL_REGION --group-id
"$CENTRAL_SG_ID" --protocol "all" --cidr "$WORKLOAD_VPC_CIDR"
```

7. Similarly, configure a workload cluster VPC security group to accept traffic from the central cluster VPC.

- a. Create a security group on the workload cluster VPC.

```
WORKLOAD_SG_ID=$(aws ec2 create-security-group --region $WORKLOAD_REGION --vpc-id
$WORKLOAD_VPC_ID --group-name central-to-workload --description "Allow central to
workload" | jq -r .GroupId) && echo $WORKLOAD_SG_ID
```

- b. Add a rule to allow incoming traffic from the central cluster VPC.

```
aws ec2 authorize-security-group-ingress --region $WORKLOAD_REGION --group-id
"$WORKLOAD_SG_ID" --protocol "all" --cidr "$CENTRAL_VPC_CIDR"
```

8. Configure the distributed workload cluster to allow access from the central cluster.

There are a few ways to do this. We'll configure the `aws-auth` config map to add a mapping for the central cluster's EC2 instance role. This allows all workloads on the central cluster to access the workload cluster.

If you want to have finer grain access control and only grant access to the necessary CN2 components, then you can use fine-grained IAM roles for service accounts. With this approach, you

would create new roles for the central cluster's contrail-k8s-deployer and contrail-k8s-kubemanager service accounts and map those instead. See AWS documentation (<https://aws.amazon.com/blogs/opensource/introducing-fine-grained-iam-roles-service-accounts/>) on how to configure IAM roles for service accounts.

- a. Get the central cluster role and store into a variable for later use.

```
CENTRAL_NODE_ROLE=$(eksctl get iamidentitymapping --cluster cn2-m-central --region
$CENTRAL_REGION -o json | jq -r .[].rolearn) && echo $CENTRAL_NODE_ROLE
```

- b. Create a mapping for the central cluster's role.

```
eksctl create iamidentitymapping --cluster cn2-m-dist --region=$DIST_REGION --arn
$CENTRAL_NODE_ROLE --group system:masters
```

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

- a. Create the secret that describes the distributed workload cluster.

Store the following into a yaml file. In this example, we call it **access-workload.yaml**.

```
kind: EKS
config:
  aws_region: us-east-2
  eks_cluster_name: cn2-m-workload
```

Create the secret:

```
kubectl --context=<central-cluster-context> create secret generic access-workload -n
contrail --from-file=kubeconfig=access-workload.yaml
```

- b. 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:
    replicas: 1
    containers:
      - image: <contrail-image-repository>
        name: contrail-k8s-kubemanager
    podV4Subnet: <pod-v4-subnet-of-remote-cluster>
    serviceV4Subnet: <service-v4-subnet-of-remote-cluster>
    clusterName: <worker-cluster-name>
    kubeconfigSecretName: <secret-name>
    listenerPort: <listener-port>
    constantRouteTargetNumber: <rt-number>

```

[Table 9 on page 58](#) explains the parameters that you need to set.

Table 9: Kubemanager CRD

Parameter	Meaning	Example
name	The name of the custom resource.	kubemanager-cluster1
image	The repository where you pull images	enterprise-hub.juniper.net/ contrail-container-prod/contrail- k8s-kubemanager:23.2.0.156
podV4Subnet	The IPv4 pod subnet that you want to use for the distributed workload cluster. This subnet must be unique within the entire multi-cluster.	10.111.64.0/18
serviceV4Subnet	The IPv4 service subnet that you configured earlier for the distributed workload cluster. This subnet must be unique within the entire multi-cluster.	172.30.0.0/16
clusterName	The name of the workload cluster.	cn2-m-workload

Table 9: Kubemanager CRD (Continued)

Parameter	Meaning	Example
kubeconfigSecretName	The name of the secret that you just created for the workload cluster. NOTE: This secret is not derived from the kubeconfig as the name suggests.	access-workload
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

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

```
kubectl --context=<central-cluster-context> apply -f kubemanager-cluster1.yaml
```

10. Install CN2 components on the distributed workload cluster.

- a. Create the secret that describes the central cluster.

Store the following into a yaml file. In this example, we call it **access-central.yaml**.

```
kind: EKS
config:
  aws_region: us-west-2
  eks_cluster_name: cn2-m-central
```

Create the contrail-deploy namespace:

```
kubectl --context=<workload-cluster-context> create ns contrail-deploy
```

Create the secret:

```
kubectl --context=<workload-cluster-context> create secret generic access-central -n  
contrail-deploy --from-file=kubeconfig=access-central.yaml
```

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

```
kubectl --context=<workload-cluster-context> apply -f manifests/  
distributed_cluster_deployer_example.yaml
```

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

```
kubectl apply --context=<workload-cluster-context> -f manifests/  
distributed_cluster_certmanager_example.yaml
```

- d. On the workload cluster, add the vRouter.

```
kubectl apply --context=<workload-cluster-context> -f manifests/  
distributed_cluster_vrouter_example.yaml
```

- e. On the workload cluster, add the worker nodes (node group).

```
eksctl create nodegroup --region $WORKLOAD_REGION --cluster cn2-m-workload --node-type m5.large --node-  
ami-family AmazonLinux2 --max-pods-per-node 100 --node-private-networking --node-security-groups  
$WORKLOAD_SG_ID
```

- f. Verify that all pods are up. This may take a few minutes.

```
kubectl get pods -A
```

11. Check over the installation.

- a. On the central cluster, verify that you can see the distributed workload cluster's namespaces.

```
kubectl --context=<workload-cluster-context> get ns
```

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

```
kubemanager-workload1-cn2-m-workload-contrail
```

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

5

CHAPTER

Appendix

[Configure Repository Credentials](#) | 63

[Juniper CN2 Technology Previews \(Tech Previews\)](#) | 64

Configure Repository Credentials

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

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

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

Enter your login credentials when prompted.

Once you've logged in, your credentials are automatically stored in `~/.docker/config.json`. (If you installed docker using snap, then the credentials are stored in the `~/snap/docker` directory hierarchy.)

3. Encode your credentials in base64 and store the resulting string.

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

Take a look at the encoded credentials.

```
echo $ENCODED_CREDS
```

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

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

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

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

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

You should see the encoded credentials in the manifests.

RELATED DOCUMENTATION

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

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.