

Juniper Paragon Automation 2.4.0 Installation and Upgrade Guide

Published
2025-11-06

RELEASE
2.4.0

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.

Juniper Paragon Automation 2.4.0 Installation and Upgrade Guide
2.4.0

Copyright © 2025 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

About This Guide | vi

1

Introduction

Paragon Automation Installation Overview | 2

2

System Requirements

Paragon Automation Implementation | 7

Paragon Automation System Requirements | 10

3

Install the Cluster

Install Paragon Automation | 22

Prepare the Cluster Nodes | 24

Download the OVA File | 24

Create the Node VMs | 27

On ESXi 8.0 | 27

On KVM | 28

On Proxmox VE | 36

Deploy the Cluster | 40

Configure the node VMs | 40

Deploy the cluster | 43

Log in to the Web GUI | 55

Post Installation Tasks | 56

Install User Certificates | 57

Update Victoria Metrics | 59

4

Upgrade the Cluster

Upgrade Paragon Automation | 62

Upgrade Prerequisites | 63

Upgrade the Paragon Automation cluster | 64

 Upgrade using the local filename Option | 65

 Upgrade using the remote url Option | 67

Upgrade Paragon Shell and the OVA System Files | 70

Post Cluster Upgrade Tasks | 71

Preupgrade Paragon Shell Before Upgrade | 72

Upgrade Prerequisites | 73

Perform Preupgrade | 74

Upgrade the Paragon Automation Cluster | 75

 Upgrade using the local Option | 75

 Upgrade using the remote url Option | 78

Upgrade Paragon Shell and the OVA System Files | 82

Post Cluster Upgrade Tasks | 83

5

Manage the Cluster

Update the OS | 85

Repair or Replace Cluster Nodes | 87

 Repair Nodes | 88

 Replace Faulty Nodes | 91

Shut Down and Reboot Nodes | 95

Increase TimescaleDB PVC Size | 97

 Increase Ceph Storage | 97

 Update Timescale DB PVC Quota | 100

 Increase Timescale DB PVC | 100

6

Backup and Restore

Back Up and Restore Paragon Automation | 103

 Back Up Using Paragon Shell | 103

 Restore Using Paragon Shell | 106

View or Delete Backup Files | 108

Upload or Download Backup Files | 109

Disaster Recovery using VMware ESXi | 110

Disaster Recovery | 110

Back Up and Restore Using VMware Snapshots | 111

Back Up Using VMware Snapshots | 111

Restore Using VMware Snapshots | 112

About This Guide

Use this guide to install Paragon Automation on one or more hypervisors. This guide explains how to:

- Install and upgrade Paragon Automation.
- Update the base-OS
- Shut down and reboot the cluster.
- Repair and replace nodes.
- Back up and restore the configuration.

RELATED DOCUMENTATION

[Juniper Paragon Automation User Guide](#)

[Juniper Paragon Automation Release Notes](#)

[Juniper Paragon Automation Monitoring and Troubleshooting Guide](#)

1

CHAPTER

Introduction

IN THIS CHAPTER

- [Paragon Automation Installation Overview | 2](#)
-

Paragon Automation Installation Overview

Juniper® Paragon™ Automation is a WAN automation solution that enables enterprise and service provider networks to meet the challenges posed by an increase in volume, velocity, and types of traffic. Paragon Automation delivers an experience-first and automation-driven network that provides a high-quality experience to network operators.

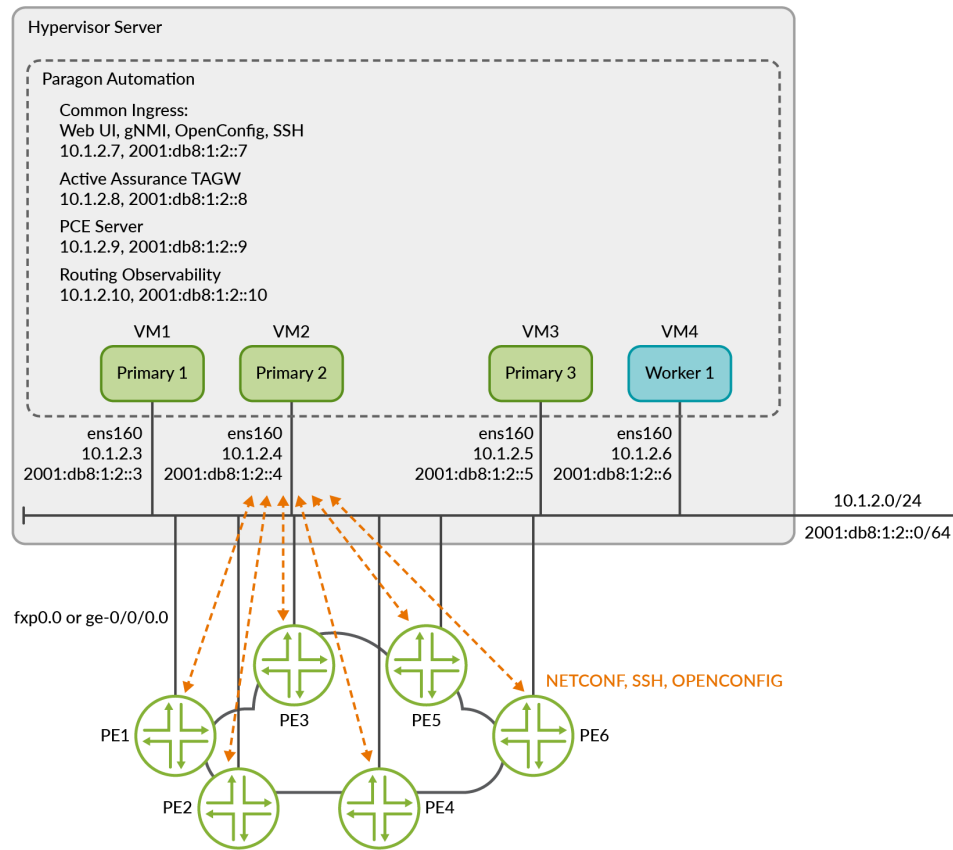
This guide describes how to install Paragon Automation and is intended for system administrators and network operators who install and manage the network infrastructure.

You deploy Paragon Automation as a set of on-premises (customer managed) nodes or virtual machines (VMs). A Kubernetes cluster is deployed inside these VMs during the installation of Paragon Automation. The Kubernetes cluster is a collection of microservices that interact with one another through APIs and that gets created automatically during Paragon Automation installation.

Inter-node communication within the cluster is implemented using APIs, and SSH, while the communication between Paragon Automation and the managed devices uses SSH, NETCONF, OpenConfig, and gNMI.

[Figure 1 on page 3](#) shows a typical Paragon Automation cluster deployment along with communication protocols. While the illustration shows two servers, you can deploy the cluster on a single server as well.

Figure 1: Paragon Automation Deployment



Paragon Automation Installation

To install Paragon Automation:

1. Download the installation bundle to your local desktop. The installation bundle comprises an OVA file. Use the OVA directly or extract the OVF and **.vmdk** files to create your VMs.
2. Create and configure the VMs on your bare-metal hypervisor using the OVA (or OVF and **.vmdk**) bundle.
3. Deploy a Paragon Automation cluster on the VMs using the Paragon Shell CLI.
4. Log in to the Paragon Automation Web GUI.

An IT or system administrator with permissions to create VMs in the hypervisor installs and maintains the Paragon Automation cluster. The IT or system administrator is responsible for tasks that are related to installation and administration. Paragon Automation can be deployed in an air-gap environment where there is no access to the Internet.

You do not have to create the VMs and then use the OVA (or OVF and **.vmdk**) bundle. You will be creating the VMs from the OVA (or OVF and **.vmdk**) bundle. In other words, you do not need to install the VMs with any particular operating system, deploy additional components such as Docker, create and configure the interfaces, configure NTP, and so on, separately. All these tasks are done automatically as part of the VM-creation process from the OVA (or OVF and **.vmdk**) bundle.

Paragon Shell CLI

Paragon Automation provides a custom containerized MGD (cMGD) user shell, called Paragon Shell. A system administrator can use Paragon Shell to deploy and configure the Paragon Automation cluster. The Paragon Shell CLI is installed and available after the VMs are created on the hypervisor server using the OVA (or OVF and **.vmdk**) bundle. The software bundle is prepackaged with all the packages that are required to create the node VMs and deploy the Paragon Automation cluster. Paragon Shell is installed on the Linux base OS.

You can use Paragon Shell to:

- Deploy the Paragon Automation cluster.
- Upgrade, back up, restore, and edit the cluster configuration.
- Create and edit users.
- Configure monitoring with the collection of metrics from different types of sources and forward the collected data to designated sinks or destinations.
- Retrieve cluster information for troubleshooting.

VMs are appliances containing both the Linux base OS as well as all required application code. When you create and log in to the VMs, you are placed in Paragon Shell, by default. When you exit Paragon Shell, you are placed in the Linux root shell.



NOTE: Exercise caution while executing commands from the Linux root shell. Commands executed from the Linux root shell are not supported unless explicitly mentioned in the documentation.

The configuration files used to deploy the cluster are stored in the **/root/epic/config** folder on the VM from which you deployed the cluster.

This guide explains how to:

- Install and upgrade Paragon Automation.
- Shut down and reboot the cluster.
- Repair and replace nodes.

- Back up and restore a configuration.

RELATED DOCUMENTATION

[Paragon Automation Implementation | 7](#)

[Paragon Automation System Requirements | 10](#)

[Install Paragon Automation | 22](#)

2

CHAPTER

System Requirements

IN THIS CHAPTER

- [Paragon Automation Implementation | 7](#)
 - [Paragon Automation System Requirements | 10](#)
-

Paragon Automation Implementation

To determine the resources required to implement Paragon Automation, you must understand the fundamentals of the Paragon Automation underlying infrastructure.

Paragon Automation is a collection of microservices that interact with one another through APIs and run within containers in a Kubernetes cluster. A Kubernetes cluster is a set of nodes or virtual machines (VMs) running containerized applications.

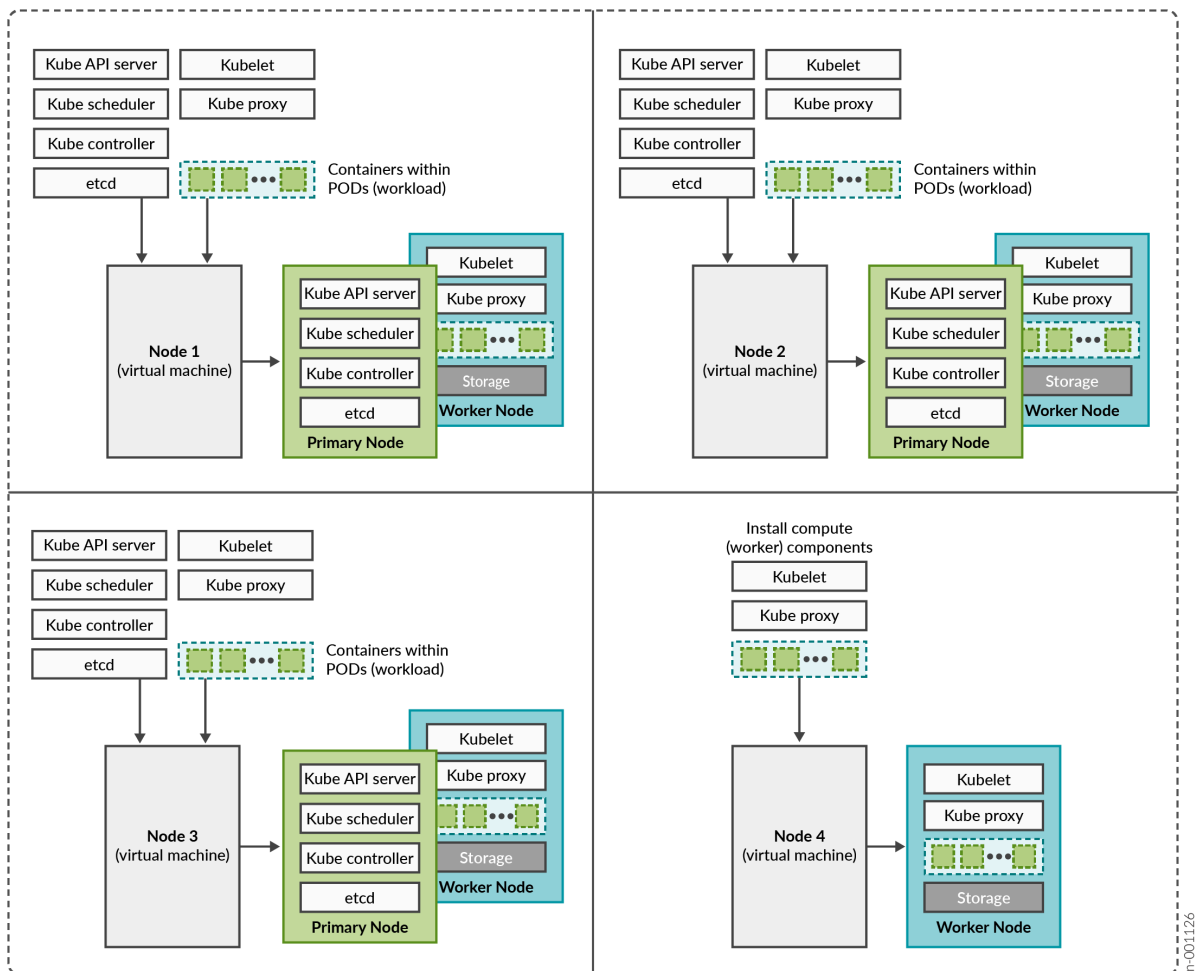
A Kubernetes cluster comprises one or more primary and worker nodes.

- **Control plane (primary) node**—The primary node performs the Kubernetes control-plane functions.
- **Compute (worker) node**—The worker node provides resources to run the pods. Worker nodes do not have control-plane function.

The two types of nodes can be deployed separately or co-located in the same VM. A single node can function as both primary and worker if the components required for both roles are installed in the same node.

In Paragon Automation, by default, the primary nodes also serve as worker nodes.

Figure 2: Kubernetes Cluster Nodes and Roles



You need to consider the intended system's capacity (number of devices to be managed, use cases, and so on), the level of availability required, and the expected system's performance, to determine the following cluster parameters:

- Total number of nodes in the cluster
- Amount of resources on each node (CPU, memory, and disk space)
- Number of nodes acting as primary and worker nodes

The amount of resources on each node are described later in this guide in "[Paragon Automation System Requirements](#)" on page 10.

Paragon Automation Implementation

Paragon Automation is implemented on top of a Kubernetes cluster, which consists of one or more primary nodes and one or more worker nodes. Paragon Automation is implemented as a multinode cluster. At minimum, three nodes that function as both primary and worker nodes and one node that functions as a worker-only node is required for a functional cluster.



NOTE: The four-node cluster is the recommended and supported implementation.

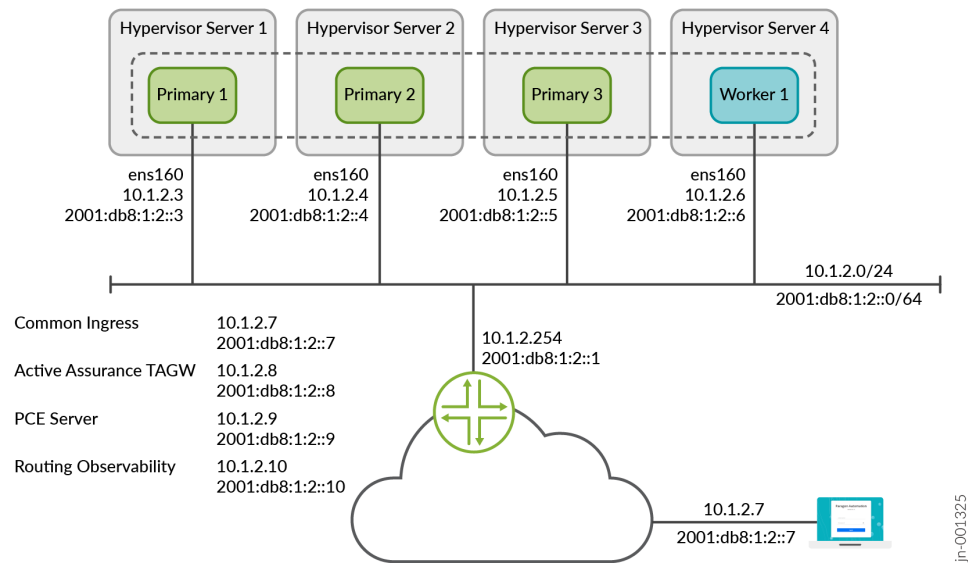
This implementation not only improves performance but allows for high availability within the cluster:

- **Control plane high availability**—The three nodes that function as both primary and worker nodes provide the required control plane redundancy. We do not support more than three primary nodes.
- **Workload high availability**—For workload high availability and workload performance, you must have more than one worker. In Paragon Automation, the three nodes that function as both primary and worker nodes and the one node that serves as a worker-only node provide the required workload high availability.
- **Storage high availability**—For storage high availability, all the nodes provide Ceph storage.

The Paragon Automation cluster remains functional when a single node fails and when the maximum latency between nodes is **less than 25 ms**.

To ensure that the cluster remains functional when a server fails, implement the cluster on four servers. The recommended implementation to maintain both server and node high availability is illustrated in [Figure 3 on page 10](#). This ensures that the cluster remains functional even if one of the servers fail.

Figure 3: Server and Node High Availability



The illustration displays a cluster where the cluster nodes and VIP addresses are all in the same subnet. The cluster nodes and VIP addresses can also be located in different subnets.

RELATED DOCUMENTATION

[Paragon Automation System Requirements | 10](#)

[Install Paragon Automation | 22](#)

Paragon Automation System Requirements

IN THIS SECTION

- [Software Requirements | 11](#)
- [Hardware Requirements | 11](#)
- [Network Requirements | 12](#)
- [Firewall Requirements | 16](#)

Before you install the Paragon Automation software, ensure that your system meets the requirements that we describe in these sections.

Software Requirements

You can deploy Paragon Automation on one or more servers of the following bare-metal hypervisors:

- VMware ESXi 8.0
- Red Hat Enterprise Linux (RHEL) 8.10 kernel-based virtual machines (KVMs).

You must install the libvirt, libvirt-daemon-kvm, bridge-utils, and qemu-kvm packages. The hypervisor must have an Intel-based CPU.

- Proxmox VE

Hardware Requirements

This section describes the minimum hardware resources that are required on each node virtual machine (VM) in the Paragon Automation cluster, for evaluation purposes or for small deployments.

The compute, memory, and disk requirements of the cluster nodes can vary based on the intended capacity of the system. The intended capacity depends on the number of devices to be onboarded and monitored, types of sensors, and frequency of telemetry messages. If you increase the number of devices, you'll need higher CPU and memory capacities.



NOTE: To get a scale and size estimate of a production deployment and to discuss detailed dimensioning requirements, contact your Juniper Partner or Juniper Sales Representative. Juniper Paragon Automation Release 2.4.0 supports a scale of a maximum of 1500 devices.

The bare minimum resources required for each of the four nodes in the cluster are:

- 16-core vCPU

- 32-GB RAM
- 512-GB SSD. SSDs are mandatory.

To configure routing observability features as well as the AI/ML (artificial intelligence [AI] and machine learning [ML]) feature to automatically monitor Key Performance Indicators (KPIs) related to a device's health, the bare minimum resources required for each of the four nodes in the cluster are:

- 48-core vCPU
- 96-GB RAM
- 2000-GB SSD



WARNING: These are the bare minimum requirements to configure routing observability and AI/ML features. To get an estimate of the resources required to configure these features on your production deployments, contact your Juniper Partner or Juniper Sales Representative.

The servers must have enough CPU, memory, and disk space to accommodate the hardware resources listed in this section. For node and server high-availability, deploy the four VMs on four servers.

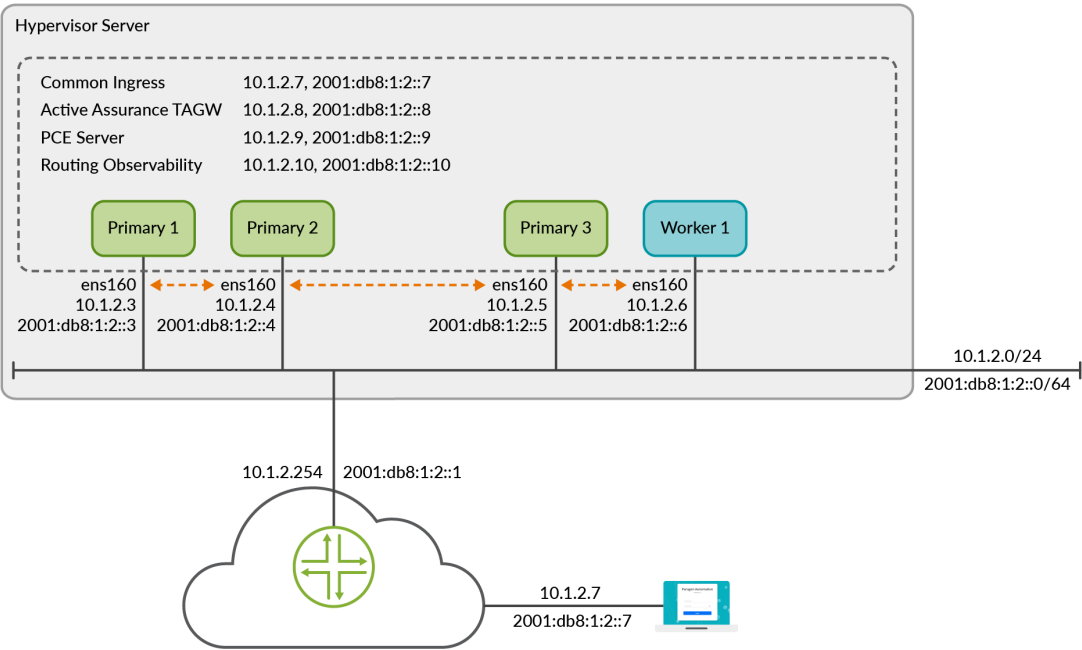
Network Requirements

The four nodes must be able to communicate with each other through SSH. The nodes must be able to sync to an NTP server. SSH is enabled automatically during the VM creation, and you will be asked to enter the NTP server address during the cluster creation. Ensure that there is no firewall blocking NTP or blocking SSH traffic between the nodes in case they are on different servers.

Single-subnet Cluster

The cluster nodes and VIP addresses can all be on the same subnet with L2 connectivity between them. [IP Addressing Requirements in a Single Subnet Cluster on page 13](#) illustrates the IP and VIP addresses within the same subnet required to install a Paragon Automation cluster.

Figure 4: IP Addressing Requirements in a Single Subnet Cluster



Multi-subnet Cluster

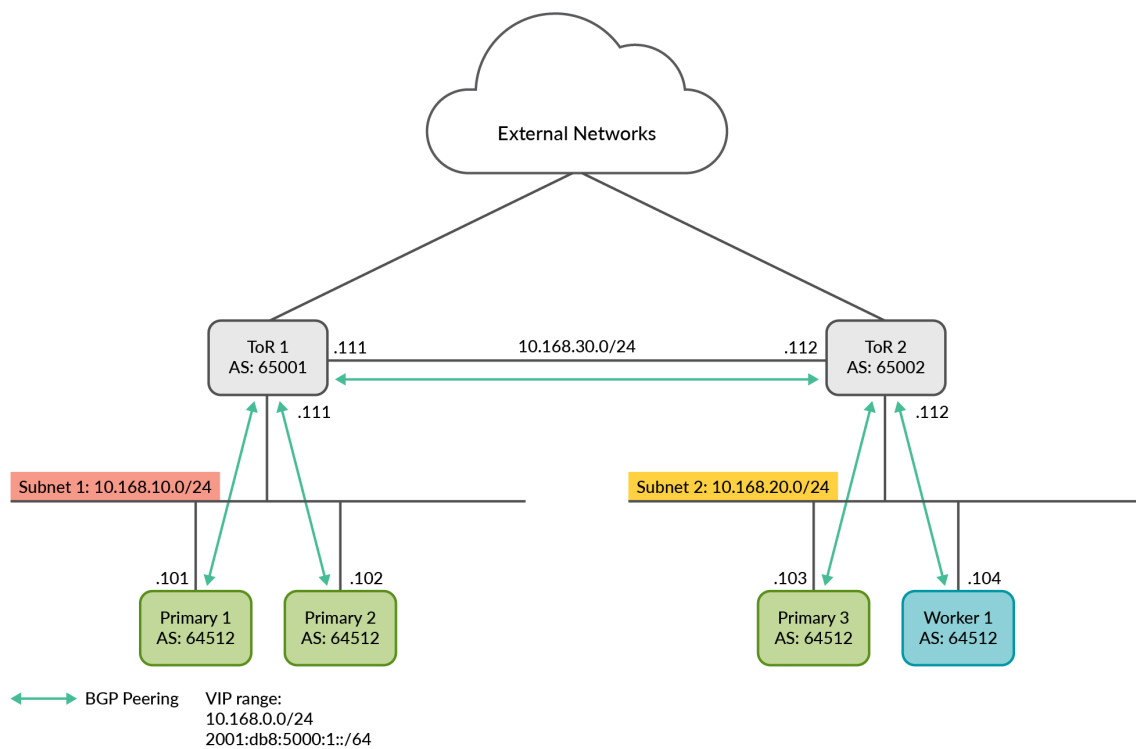
Alternatively, in cases where the cluster nodes are geographically distributed or are located in multiple data centers, the nodes and the VIP addresses can be in different subnets. You must configure BGP peering between each cluster node and the respective upstream gateway top-of-rack (ToR) router as well as between the routers. Additionally, the cluster nodes must have the same configured AS number.



NOTE: BGP connectivity configuration between your routers and VMs is beyond the scope of this document. You will need to ensure BGP peering is established between the cluster node VMs and the ToR routers.

Figure 5 on page 14 illustrates a cluster in two different networks. Two nodes are served by ToR1 router and two nodes are served by ToR2 router. In this example, you must configure EBGW using interface peering between ToR1 and Primary1 and Primary2, and between ToR2 and Primary3 and Primary4, as well as between ToR1 and ToR2. Your BGP configuration might differ based on your setup.

Figure 5: Multi-subnet Cluster



Configure IPv4 Addresses

You need to have the following IP addresses available for the installation.

- Four interface IP addresses, one for each of the four nodes
- Internet gateway IP address
- Virtual IP (VIP) addresses for:
 - Generic ingress IP address shared between gNMI, NETCONF (SSH connections from devices), and the Web GUI—This is a general-purpose VIP address that is shared between multiple services and used to access Paragon Automation from outside the cluster.

Alternatively, in cases where the device management network is on a different subnet than the network used to access the GUI, you can also use one VIP address for the Web GUI and a separate VIP address for gNMI and NETCONF access.

If an additional IP address is defined (using `ingress ingress-vip` option) and configured (using `oc-term oc-term-host` and `gnmi gnmi-term-host` options), the additional IP address is added to the outbound SSH configuration used to adopt devices from that network. The VIP address from the first

network is ideally used to access the GUI. While both sets of IP addresses can be used to access the GUI, NETCONF, and gNMI, only the defined and configured address is added to the outbound SSH configuration for NETCONF and gNMI. If you want to adopt devices from the first subnet, you must manually edit the outbound SSH command to over-write the configured IP address for NETCONF and gNMI access.

- Paragon Active Assurance Test Agent gateway (TAGW)—This VIP address serves HTTP-based traffic to the Paragon Active Assurance Test Agent endpoint.
- PCE server—This VIP address is used to establish Path Computational Element Protocol (PCEP) sessions between Paragon Automation and the devices. The PCE server VIP configuration is necessary to view dynamic topology updates in your network in real-time. For information on establishing BGP-LS peering and PCEP sessions, see *Dynamic Topology Workflow*.
- Routing observability cRPD—This VIP is used by external network devices as BGP Monitoring Protocol (BMP) station IP address to establish the BMP session.

The VIP addresses are added to the outbound SSH configuration that is required for a device to establish a connection with Paragon Automation.



NOTE: In a multi-subnet cluster installation, the VIP addresses must *not* be on the same subnet as the cluster nodes.

- Hostnames mapped to the VIP addresses—Along with VIP addresses, you can also enable devices to connect to Paragon Automation using hostnames. However, you must ensure that the hostnames and the VIP addresses are correctly mapped in the DNS and your device is able to connect to the DNS. If you configure Paragon Automation to use hostnames, the hostnames take precedence over VIP addresses and are added to the outbound SSH configuration used during onboarding devices.

Configure IPv6 Addresses

You can configure the Paragon Automation cluster using IPv6 addresses in addition to the existing IPv4 addresses. With IPv6 addressing configured, you can use IPv6 addresses for NETCONF, gNMI, the Active Assurance TAGW, and access to the Web GUI. You must have the following additional addresses available at the time of installation:

- Four interface IPv6 addresses, one for each of the four nodes
- Internet gateway IPv6 address
- One IPv6 VIP address for generic ingress or two IPv6 VIP addresses, one for the Web GUI and one for NETCONF and gNMI access
- One IPv6 VIP address for Active Assurance TAGW

- Hostnames mapped to the IPv6 VIP addresses—You can also use hostnames to connect to IPv6 addresses. You must ensure that the hostnames are mapped correctly in the DNS to resolve to the IPv6 addresses.

If hostnames are not configured and IPv6 addressing is enabled in the cluster, the IPv6 VIP addresses are added to the outbound SSH configuration, used for device onboarding, instead of IPv4 addresses.

You must configure the IPv6 addresses at the time of cluster deployment. You cannot configure IPv6 addresses after a cluster has been deployed using only IPv4 addresses.



NOTE: We do not support configuring an IPv6 address for the PCE server and the routing observability feature.

In addition to the listed IP addresses and hostnames, you need to have the following information available with you at the time of installation:

- Primary and secondary DNS server addresses for IPv4 and IPv6 (if needed)
- NTP server information

Firewall Requirements

The following section lists the ports that firewalls must allow for communication within and from outside of the cluster.

You must allow intracluster communication between the nodes. In particular, you must keep the ports listed in [Table 1 on page 16](#) open for communication.

Table 1: Ports That Firewalls Must Allow for Intracluster Communication

Port	Usage	From	To	Comments
Infrastructure Ports				
22	SSH for management	All cluster nodes	All cluster nodes	Require a password or SSH-key
2222 TCP	Paragon Shell configuration sync	All cluster nodes	All cluster nodes	Require password or SSH-key

Table 1: Ports That Firewalls Must Allow for Intracluster Communication (*Continued*)

Port	Usage	From	To	Comments
443 TCP	HTTPS for registry	All cluster nodes	Primary nodes	Anonymous read access Write access is authenticated
2379 TCP	etcd client port	Primary nodes	Primary nodes	Certificate-based authentication
2380 TCP	etcd peer port	Primary nodes	Primary nodes	Certificate-based authentication
5473	Calico CNI with Typha	All cluster nodes	All cluster nodes	—
6443	Kubernetes API	All cluster nodes	All cluster nodes	Certificate-based authentication
7472 TCP	MetallB metric port	All cluster nodes	All cluster nodes	Anonymous read only, no write access
7946 UDP	MetallB member election port	All cluster nodes	All cluster nodes	—
8443	HTTPS for registry data sync	Primary nodes	Primary nodes	Anonymous read access Write access is authenticated
9345	rke2-server	All cluster nodes	All cluster nodes	Token based authentication

Table 1: Ports That Firewalls Must Allow for Intracluster Communication (*Continued*)

Port	Usage	From	To	Comments
10250	kubelet metrics	All cluster nodes	All cluster nodes	Standard Kubernetes authentication
10260	RKE2 cloud controller	All cluster nodes	All cluster nodes	Standard Kubernetes authentication
32766 TCP	Kubernetes node check for PCE service local traffic policy	All cluster nodes	All cluster nodes	Read access only
Calico CNI Ports				
4789 UDP	Calico CNI with VXLAN	All cluster nodes	All cluster nodes	—
5473 TCP	Calico CNI with Typha	All cluster nodes	All cluster nodes	—
51820 UDP	Calico CNI with Wireguard	All cluster nodes	All cluster nodes	—

The following ports must be open for communication from outside the cluster.

Table 2: Ports That Firewalls Must Allow for Communication from Outside the Cluster

Port	Usage	From	To
179 TCP	Topology visualization and traffic engineering using the topology information	Paragon cluster node IP address	Router IP address to which you want to set up BGP peering from Paragon Automation. You can use the router management IP address or the router interface IP address.
443	Web GUI + API	External user computer/desktop	Web GUI Ingress VIP address(es)
443	Paragon Active Assurance Test Agent	External network devices	Paragon Active Assurance Test Agent VIP address
2200	NETCONF	External network devices	Web GUI Ingress VIP address(es)
4189	PCE Server	External network devices	PCE Server VIP address
6800	Paragon Active Assurance Test Agent	External network devices	Paragon Active Assurance Test Agent VIP address
32767	gNMI	External network devices	Web GUI Ingress VIP address(es)
17002	Routing Observability	External network devices	Routing observability cRPD load balancer IP address

Web Browser Requirements

The latest versions of Google Chrome, Mozilla Firefox, and Safari.



NOTE: We recommend that you use Google Chrome.

RELATED DOCUMENTATION

[Paragon Automation Implementation | 7](#)

[Install Paragon Automation | 22](#)

3

CHAPTER

Install the Cluster

IN THIS CHAPTER

- [Install Paragon Automation | 22](#)
 - [Prepare the Cluster Nodes | 24](#)
 - [Deploy the Cluster | 40](#)
 - [Log in to the Web GUI | 55](#)
 - [Post Installation Tasks | 56](#)
-

Install Paragon Automation

SUMMARY

This topic describes at a high-level the steps that you must perform to install Paragon Automation on a hypervisor using an OVA file.

You (system administrator) can install Paragon Automation by downloading an OVA bundle. Use the OVA bundle to deploy the node virtual machines (VMs) on one or many servers. Alternatively, extract the OVF and `.vmdk` files from the OVA bundle and use the files to deploy the node VMs. Paragon Automation runs on a Kubernetes cluster with at least three primary and worker nodes and one worker-only node. The installation is air-gapped but you need Internet access to download the OVA bundle to your computer.

If you are installing from a remote machine, you can instead create a local desktop installer VM, either on the same server where you want to install Paragon Automation or on a different server. The local desktop installer VM can be a basic Ubuntu desktop VM. This avoids having to download the files into your remote machine, and then running the installation from the remote machine. This is described in the ["download the OVA" on page 24](#) section.

To install Paragon Automation, you must create the node VMs using the downloaded OVA (or OVF and `.vmdk`) files. The software download files come prepackaged with the base OS and all packages required to create the VMs and deploy your Paragon Automation cluster. The VMs have Ubuntu 22.04.5 LTS (Jammy Jellyfish) Linux base OS.

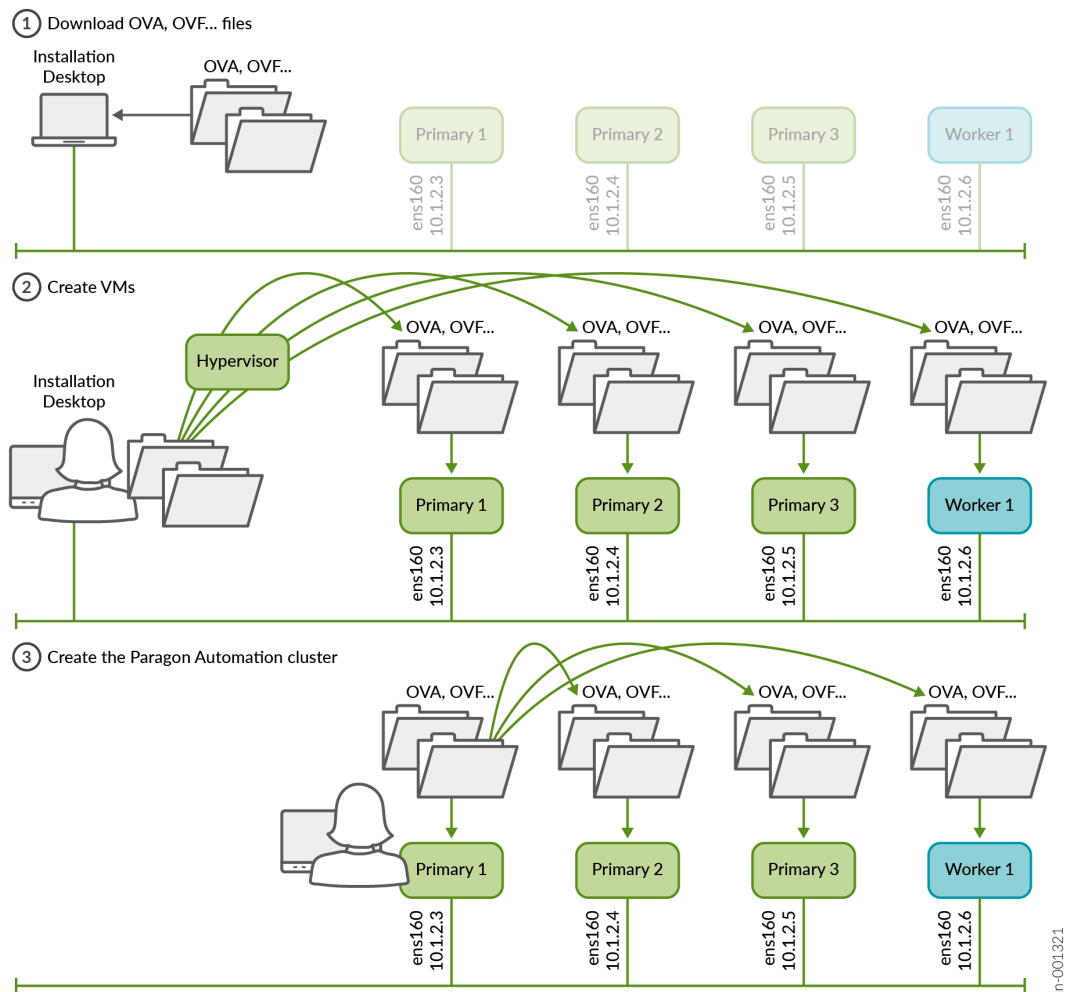
Once the VMs are created, you must configure each VM in the same way. When all the VMs are configured and prepared, you deploy the Paragon Automation cluster. You can deploy the cluster using Paragon Shell.

To install Paragon Automation, perform the following steps:

1. ["Prepare the nodes" on page 24](#)—Download the OVA and create the node VMs.
2. ["Deploy the cluster" on page 40](#)—Configure the node VMs and deploy the cluster.
3. ["Log in to the Web GUI" on page 55](#)—Verify the deployment and log in to the Web GUI.

[Figure 6 on page 23](#) illustrates the Paragon Automation installation process at a high-level.

Figure 6: Installation Process



NOTE: You create the VMs directly using the OVA (or OVF and **.vmdk**) bundle. You do not need to create the VMs separately with any running software (for example, Ubuntu), or explicitly create interfaces, install Docker separately and so on. These are all automatically created and configured when you create the VMs using the OVA (or OVF and **.vmdk**) bundle.

Go to ["Prepare the nodes" on page 24](#) to begin the installation process.

RELATED DOCUMENTATION

[Paragon Automation System Requirements](#) | 10

Prepare the Cluster Nodes

SUMMARY

This topic describes the steps you must perform to prepare the Paragon Automation cluster nodes for deployment.

IN THIS SECTION

- [Download the OVA File | 24](#)
- [Create the Node VMs | 27](#)

To prepare your cluster nodes for deployment on ESXi 8.0, KVM, and Proxmox VE hypervisors, perform the following steps:

1. ["Download the OVA File" on page 24.](#)
2. ["Create the Node VMs" on page 27.](#)

The creation method differs based on the bare-metal hypervisor on which you want to deploy the cluster.

Download the OVA File

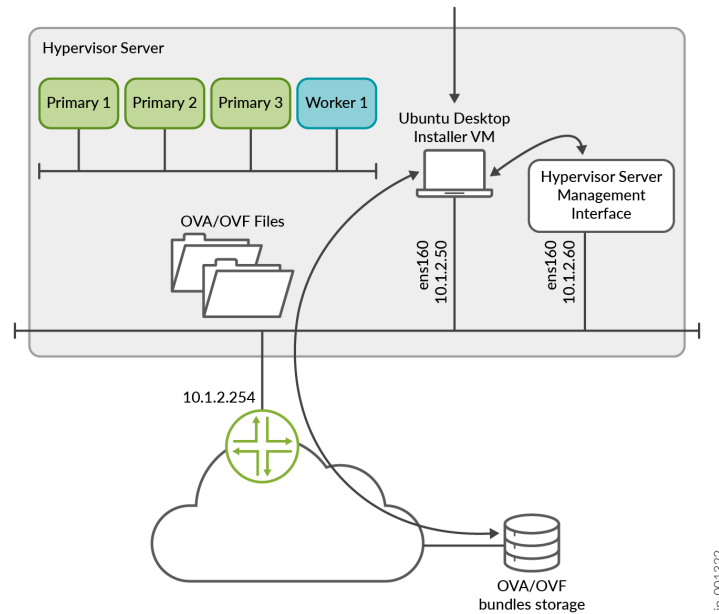
Download and verify the integrity of the OVA file.

1. Download the **Paragon Automation Installation OVA** file from the Juniper Paragon Automation [software download site](#). The OVA is used to create the node VMs and deploy your cluster.

Note that the actual filename will include the release date in it, such as **paragon-2.4. X-builddate.ova**.

The file is large in size, and it might take considerable time to download it and then create the VMs from your computer. So, we recommend that you create a local installer VM, which can be a basic Ubuntu desktop VM, either on the same server where you want to install Paragon Automation or on a different server. You must be able to download the OVA file to this local installer VM and you must have enough space on the VM to store the file. Configure connectivity to the management IP addresses of the servers as show in [Figure 7 on page 25](#).

Figure 7: Local Installer VM to download the OVA/OVF files



Alternatively, you can use the `wget "http://cdn.juniper.net/software/file-download-url/paragon-2.4.X-builddate.ova"` command to download the OVA directly on to your hypervisor.

2. (Optional) Validate the integrity of the OVA file. If you are using an Ubuntu desktop, use the following command:

```
root@ubuntu:~$ sha512sum paragon-2.4.  
X-builddate.ova  
7deda68aae8ba6399aa95d5365a659a8d579c5562811ebe588972cf0c5107337628370d78dcdb56ab8ea97e73b759  
7f3a5ff06e9f501706bd8954b7454b86d2 paragon-2.4.  
X-builddate.ova
```

Verify that the number displayed onscreen is the same as the SHA512 checksum number available on the Juniper Paragon Automation [software download site](#). Click **Checksums** to view the valid SHA512 checksum.

3. On ESXi 8.0

If you are using ESXi 8.0, you can use the OVA directly to create the VMs.

You can also extract and use the OVF and **.vmdk** files from the OVA to create your VMs. To extract the files, use the following command:

```
root@ubuntu:~# tar -xvf paragon-2.4.X-builddate.ova
paragon-2.4.X-builddate-disk1.vmdk
paragon-2.4.X-builddate-disk2.vmdk
paragon-2.4.X-builddate-file1.nvram
paragon-2.4.X-builddate.mf
paragon-2.4.X-builddate.ovf
```

If your installation desktop is running Windows, you can download and use the tar utility from <https://gnuwin32.sourceforge.net/packages/gtar.htm> to extract the files.



NOTE: If you are using a stand-alone ESXi 8.0 server without vCenter, due to a limitation of the VMware host client, you cannot upload large OVA files to the client. In such cases, you must extract and use the OVF, **.vmdk**, and **.nvram** files to create your VMs.

On KVM and Proxmox VE

If you are using KVM, you must extract the **.vmdk** files from the OVA using the `tar -xvf paragon-2.4.X-builddate.ova` command.

The rest of this document assumes that the OVA is downloaded to a single KVM server. If you have multiple servers, then download the files onto all the servers. The steps described in this document are general guidelines to create the VMs using the CLI method of deployment. You can also use GUI-based deployment and alter the steps to match your hypervisor requirements. Network configuration of the hypervisor is out of scope of this document.

Now you can create the node VMs.

Create the Node VMs

IN THIS SECTION

- On ESXi 8.0 | 27
- On KVM | 28
- On Proxmox VE | 36

After verifying the integrity of the OVA file, create the node VMs. Use one of the following methods to create the node VMs based on the hypervisor on which you are deploying the cluster.

On ESXi 8.0

Perform the following steps on ESXi 8.0 hypervisors.

1. From your Web browser, connect and log in to the VMware ESXi 8.0 server where you will install Paragon Automation.

If you are using a local installer VM, use the browser in the VM to connect to the VMware ESXi server.

2. Create the node VMs.

Perform the following steps to create the VMs.

- a. Right-click the **Host** icon and select **Create/Register VM**.

The New virtual machine wizard appears.

- b. On the Select creation type page, select **Deploy a virtual machine from an OVF or OVA file**.

Click **Next**.

- c. On the Select OVF and VMDK files page, enter a name for the node VM.

Click to upload or drag and drop the OVA file (or the OVF and **.vmdk** files).

Review the list of files to be uploaded and click **Next**.

- d. On the Select storage page, select the appropriate datastore that can accommodate 300-GB SSD for the node VM. Note that SSD is mandatory.

Click **Next**. The extraction of files takes a few minutes.

e. On the Deployment options page:

- Select the virtual network to which the node VM will be connected.
- Select the **Thick** disk provisioning option.
- Enable the VM to power on automatically.

Click **Next**.

f. On the Ready to complete page, review the VM settings.

Click **Finish** to create the node VM.



NOTE: If you used the OVF and **.vmdk** files to create your VMs and the VM creation failed, retry creating the VMs with the **.nvram** file. On step ["2.c" on page 27](#), upload the **.nvram** file along with the OVF and **.vmdk** files. For standalone ESXi 8.0 servers without vCenter, you must upload the **.nvram** file as well.

g. To power on the VM, right-click the newly created VM on the Inventory page, and click **Power > Power on**.

h. Repeat steps ["2.a" on page 27](#) through ["2.g" on page 28](#) for the other three node VMs.

Alternatively, if you are using VMware vCenter, you can right-click the VM, and click the **Clone > Clone to Virtual Machine** option to clone the newly created VM. Clone the VM three times to create the remaining node VMs.

Enter appropriate VM names when prompted.

i. (Optional) Verify the progress of the VM creation in the Recent tasks section at the bottom of the page. When a VM is created, it appears in the VMware Host Client inventory under Virtual Machines.

j. When all the VMs have been created, verify that the VMs have the correct specifications and are powered on.

You have completed the node preparation steps and created all the VMs. You are ready to configure and deploy the cluster. Go to ["Deploy the Cluster" on page 40](#).

On KVM

Perform the following steps on KVM hypervisors with RHEL 8.10 host OS.

In this example, we are deploying the cluster on a single hypervisor server with the following location and naming parameters:

- Create two VMs each in two data locations (SSDs) on the same hypervisor.

- `/data1/paragon1/` for VM1 and `/data1/paragon2/` for VM2
- `/data2/paragon3/` for VM3 and `/data2/paragon4/` for VM4

Where VM1, VM2, VM3, and VM4 are the four cluster node VMs. We recommend using two SSDs for the node VMs to optimize IOPS rate.



NOTE: While this example showcases VMs on a single hypervisor server, for server and node high availability, you must create one VM per hypervisor server.

- VM1 is named as `paragon1`, VM2 is named as `paragon2`, VM3 is named as `paragon3`, and VM4 is named as `paragon4`.
- The two disk images for each VM are named **`paragon-disk1.img`** and **`paragon-disk2.img`**. Both the disk images for each VM are located in the corresponding **`paragonx`** directory. Where *x* is the VM number (1 through 4).
- For all VMs, the CPU = 16, RAM = 32-GB, and Mode = host cpu. These are the *bare minimum* hardware resources that are required on each node VM.
- VMs are attached to the `br-ex` bridged network.

Perform the following steps to create the node VMs.

1. Log in to the KVM hypervisor CLI.
2. Ensure that you have the required `libvirt`, `libvirt-daemon-kvm`, `qemu-kvm`, `bridge-utils`, and `qemu-kvm` packages installed.

Use the `dnf update` and `dnf install libvirt libvirt-daemon-kvm qemu-kvm bridge-utils` commands to install the packages.

3. Convert the `.vmdk` files to the raw format.

```
root@kvm:~/paragon# qemu-img convert -O raw paragon-2.4.X-builddate-disk1.vmdk paragon-disk1.img
root@kvm:~/paragon# qemu-img convert -O raw paragon-2.4.X-builddate-disk2.vmdk paragon-disk2.img
root@kvm:~/paragon# ls -l
total 111327724
-rw-r--r-- 1 root root 268435456000 Feb 13 16:23 paragon-disk1.img
-rw-r--r-- 1 root root 53687091200 Feb 13 16:23 paragon-disk2.img
-rw-r--r-- 1 64 64 34630534656 Feb 5 10:08 paragon-2.4.X-builddate-disk1.vmdk
```

```
-rw-r--r-- 1 64 64 74240 Feb 5 10:08 paragon-2.4.X-builddate-disk2.vmdk
-rw-r--r-- 1 64 64 8684 Feb 5 10:08 paragon-2.4.X-builddate-file1.nvram
-rw-r--r-- 1 64 64 394 Feb 5 09:26 paragon-2.4.X-builddate.mf
-rw-r--r-- 1 root root 34679057408 Feb 5 10:08 paragon-2.4.X-builddate.ova
-rw-r--r-- 1 64 64 10866 Feb 5 09:26 paragon-2.4.X-builddate.ovf
```

Here, **paragon-disk1.img** is the main disk and **paragon-disk2.img** is the Ceph disk.

4. Adjust and expand the disk size. In this example, we have used the bare minimum hardware resources that are required on each node VM.

```
root@kvm:~/paragon# qemu-img resize -f raw paragon-disk1.img 300G
Image resized.
root@kvm:~/paragon# qemu-img resize -f raw paragon-disk2.img 75G
Image resized.
root@kvm:~/paragon# ls -l
total 111327724
-rw-r--r-- 1 root root 322122547200 Feb 13 16:23 paragon-disk1.img
-rw-r--r-- 1 root root 80530636800 Feb 13 16:23 paragon-disk2.img
-rw-r--r-- 1 64 64 34630534656 Feb 5 10:08 paragon-2.4.X-builddate-disk1.vmdk
-rw-r--r-- 1 64 64 74240 Feb 5 10:08 paragon-2.4.X-builddate-disk2.vmdk
-rw-r--r-- 1 64 64 8684 Feb 5 10:08 paragon-2.4.X-builddate-file1.nvram
-rw-r--r-- 1 64 64 394 Feb 5 09:26 paragon-2.4.X-builddate.mf
-rw-r--r-- 1 root root 34679057408 Feb 5 10:08 paragon-2.4.X-builddate.ova
-rw-r--r-- 1 64 64 10866 Feb 5 09:26 paragon-2.4.X-builddate.ovf
```

5. Create the folders where you want the VMs to be located.
 - **/data1/paragon1/** for VM1
 - **/data1/paragon2/** for VM2
 - **/data2/paragon3/** for VM3
 - **/data2/paragon4/** for VM4
6. Copy both the **paragon-disk1.img** and **paragon-disk2.img** raw disk images to the location of each VM using the **cp** copy command. For example, `root@kvm:~/paragon# cp paragon-disk1.img /data1/paragon1/`.

Once copied, the folder and file structure will be similar to:

```
root@kvm:~/paragon# ls -l /data1/paragon1
total 39852200
```

```

-rw-r--r-- 1 root root 322122547200 Feb 28 05:45 paragon-disk1.img
-rw-r--r-- 1 root root 80530636800 Feb 13 16:30 paragon-disk2.img
root@kvm:~/paragon# ls -l /data1/paragon2
total 39792188
-rw-r--r-- 1 root root 322122547200 Feb 28 05:50 paragon-disk1.img
-rw-r--r-- 1 root root 80530636800 Feb 13 16:32 paragon-disk2.img
root@kvm:~/paragon# ls -l /data2/paragon3
total 39789780
-rw-r--r-- 1 root root 322122547200 Feb 28 05:51 paragon-disk1.img
-rw-r--r-- 1 root root 80530636800 Feb 13 16:30 paragon-disk2.img
root@kvm:~/paragon# ls -l /data2/paragon4
total 39796652
-rw-r--r-- 1 root root 322122547200 Feb 28 05:51 paragon-disk1.img
-rw-r--r-- 1 root root 80530636800 Feb 13 16:29 paragon-disk2.img

```

7. Generate and customize an XML file for configuring each VM.

Configure the machine type as q35 and emulator binary as `/usr/libexec/qemu-kvm`.

A sample XML file for RHEL 8.10 is described here:

```

<domain type='kvm'>

  <!-- Specify VM name here -->
  <name>paragon1</name>

  <!-- Specify VM RAM size here -->
  <memory unit='KiB'>33554432</memory>
  <currentMemory unit='KiB'>33554432</currentMemory>

  <!-- Specify number of vcpu for the VM here -->
  <vcpu placement='static'>16</vcpu>

  <!--
  For Ubuntu 22.04 KVM use pc-q35-jammy as machine type
  For RHEL 8.10 KVM use q35 as machine type
  -->
  <os>
    <type arch='x86_64' machine='q35'>hvm</type>
  </os>

  <features>

```

```

    <acpi/>
    <apic/>
    <vmport state='off' />
</features>
<cpu mode='host-passthrough' check='none' migratable='on' />
<clock offset='utc'>
    <timer name='rtc' tickpolicy='catchup' />
    <timer name='pit' tickpolicy='delay' />
    <timer name='hpet' present='no' />
</clock>
<on_poweroff>destroy</on_poweroff>
<on_reboot>restart</on_reboot>
<on_crash>destroy</on_crash>
<pm>
    <suspend-to-mem enabled='no' />
    <suspend-to-disk enabled='no' />
</pm>
<devices>

    <!--
    For Ubuntu 22.04 KVM use /usr/bin/qemu-system-x86_64 as emulator
    For RHEL 8.10 KVM use /usr/libexec/qemu-kvm as emulator
    -->
    <emulator>/usr/libexec/qemu-kvm</emulator>

    <disk type='file' device='disk'>
        <driver name='qemu' type='raw' cache='none' discard='ignore' />
        <!-- Specify the path to the 1st virtual disk for the main disk -->
        <source file='/data1/paragon1/paragon-disk1.img' />
        <target dev='vda' bus='virtio' />
        <boot order='1' />
        <address type='pci' domain='0x0000' bus='0x05' slot='0x00' function='0x0' />
    </disk>
    <disk type='file' device='disk'>
        <driver name='qemu' type='raw' cache='none' discard='ignore' />
        <!-- Specify the path to the 2nd virtual disk for the CEPH OSD disk -->
        <source file='/data1/paragon1/paragon-disk2.img' />
        <target dev='vdb' bus='virtio' />
        <address type='pci' domain='0x0000' bus='0x06' slot='0x00' function='0x0' />
    </disk>
    <controller type='usb' index='0' model='qemu-xhci' ports='15'>
        <address type='pci' domain='0x0000' bus='0x02' slot='0x00' function='0x0' />
    </controller>

```

```

<controller type='scsi' index='0' model='virtio-scsi'>
  <address type='pci' domain='0x0000' bus='0x03' slot='0x00' function='0x0' />
</controller>
<controller type='sata' index='0'>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x1f' function='0x2' />
</controller>
<controller type='virtio-serial' index='0'>
  <address type='pci' domain='0x0000' bus='0x04' slot='0x00' function='0x0' />
</controller>
<interface type='bridge'>
  <!-- Specify the linux bridge name for the VM to attach to -->
  <source bridge='br-ex' />
  <model type='virtio' />
  <address type='pci' domain='0x0000' bus='0x01' slot='0x00' function='0x0' />
</interface>
<serial type='pty'>
  <target type='isa-serial' port='0'>
    <model name='isa-serial' />
  </target>
</serial>
<console type='pty'>
  <target type='serial' port='0' />
</console>
<channel type='unix'>
  <target type='virtio' name='org.qemu.guest_agent.0' />
  <address type='virtio-serial' controller='0' bus='0' port='1' />
</channel>
<input type='keyboard' bus='ps2' />

<!--
Specify the TCP port for VNC access for GUI console access
The port number should be uniq and unused TCP port.
KVM can also allocate dynamic port by setting port='' and autoport='yes'
-->
<graphics type='vnc' port='5911' autoport='no' listen='0.0.0.0'>
  <listen type='address' address='0.0.0.0' />
</graphics>

<video>
  <model type='qxl' ram='65536' vram='65536' vgamem='16384' heads='1' primary='yes' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x0' />
</video>
<memballoon model='virtio'>

```

```

        <address type='pci' domain='0x0000' bus='0x07' slot='0x00' function='0x0' />
    </memballoon>
    <rng model='virtio'>
        <backend model='random'>/dev/urandom</backend>
        <address type='pci' domain='0x0000' bus='0x08' slot='0x00' function='0x0' />
    </rng>
</devices>
</domain>

```

Save this file as **/root/paragon/paragon1.xml**. In this example:

- VM1 is named as paragon1.
- VM1 CPU = 16, VM1 RAM = 32-GB, VM1 Mode = host cpu
- VM1 has two image disk images at **/data1/paragon1/paragon-disk1.img** and **/data1/paragon1/paragon-disk2.img**
- The VM is attached to bridged network with name br-ex
- The VNC port for graphical console is listening on port 5911. If you want to dynamically assign ports, then set `autoport='yes'`.

8. Define the VM using the XML file.

```

root@kvm:~/paragon# virsh define paragon1.xml
Domain 'paragon1' defined from paragon1.xml

```

9. Verify that the VM is registered.

```

root@kvm:~/paragon# virsh list --all
 Id   Name      State
-----
-   paragon1  shut off

```

10. Set the VM to autostart if the KVM is rebooted.

```

root@kvm:~/paragon# virsh autostart paragon1
Domain 'paragon1' marked as autostarted

```

11. Power on the VM.

```
root@kvm:~/paragon# virsh start paragon1
Domain 'paragon1' started
```

12. Connect to the VM console in one of the following ways:

- Using the serial console.

Connect to the VM using the serial console.

```
root@kvm:~/paragon# virsh console paragon1
Connected to domain 'paragon1'
Escape character is ^] (Ctrl + ])
[ OK ] Listening on Journal Socket.
[ 6.635248] systemd[1]: Listening on Network Service Netlink Socket.
[ OK ] Listening on Network Service Netlink Socket.
[ 6.640195] systemd[1]: Listening on udev Control Socket.
[ OK ] Listening on udev Control Socket.
[ 6.646190] systemd[1]: Listening on udev Kernel Socket.
[ OK ] Listening on udev Kernel Socket.
[ 6.651480] systemd[1]: Mounting Huge Pages File System...
...
...
[ OK ] Reached target Login Prompts.
[ OK ] Reached target Multi-User System.
[ OK ] Reached target Graphical Interface.
Starting Record Runlevel Change in UTMP...
[ OK ] Finished Record Runlevel Change in UTMP.

epic login:
```

- Using a VNC client.

Use any VNC compatible client and connect to *kvm-ip*:5911.

Ensure that the firewall allows port 5911 for communication between your computer and the VM.

13. Repeat steps "7" on page 31 through "12" on page 35 for the remaining three VMs.

When you create XML files for the remaining three VMs, change the VM name and disk locations as appropriate. Also, change the graphical console port numbers for each VM.

You have completed the node preparation steps and created all the VMs. You are ready to configure and deploy the cluster. Go to ["Deploy the Cluster" on page 40](#).

On Proxmox VE

Perform the following steps on Proxmox VE hypervisors.

In this example, we are installing Paragon Automation on a single server with three provisioned datastores, data0, data1, and data2. data0 is used to save the OVA file, data1 to save the first disk image, and data2 to save the second disk image. The VMs are named, VM1, VM2, VM3, and VM4. We are also configuring the bare minimum hardware resources required to deploy a cluster.

Perform the following steps to create the node VMs.

1. Log in to the Proxmox VE server CLI.
2. Create the data0, data1, and data2 datastores.
3. Copy the OVA to the data0 datastore and extract the **.vmdk** files.

In this example, we have created a folder called **ova** in data0, copied the **paragon-2.4. X-builddate.ova** file to the **ova** folder, and used the `tar -xvf paragon-2.4. X-builddate.ova` command to extract the files.

4. Create the first VM with a VirtIO network interface (net0), and configure the VM name, ID, and memory.

```
root@proxmox:# qm create 100 --memory 32768 --name VM1 --net0 virtio,bridge=vmbr0
```

Here, the VM ID is 100, VM name is VM1, and VM memory is 32-GB.

Ensure that the VM ID is a unique identifier and is not shared by any other VM on the same server or in the same Proxmox cluster.

5. Configure the total number of vCPUs as 16.

```
root@proxmox# qm set 100 --sockets 2
update VM 100: -sockets 2
root@proxmox# qm set 100 --cores 8
update VM 100: -cores 8
```

6. Configure storage for the VM.

```
root@proxmox# qm set 100 -scsihw virtio-scsi-single
update VM 100: -scsihw virtio-scsi-single
```

7. Configure the CPU type as host.

```
root@proxmox# qm set 100 --cpu cputype=host
update VM 100: -cpu cputype=host
```

8. Navigate to the **data0/ova** folder.

9. Import disk 1 to the VM as a raw image. Here, import the **paragon-2.4.X-builddate-disk1.vmdk** file to the data1 datastore.

```
root@proxmox:/mnt/pve/data0/ova# qm importdisk 100 paragon-2.4.X-builddate-disk1.vmdk data1
importing disk 'paragon-2.4.X-builddate-disk1.vmdk' to VM 100 ...
Formatting '/mnt/pve/data1/images/100/vm-100-disk-0.raw', fmt=raw size=268435456000
preallocation=off
transferred 0.0 B of 250.0 GiB (0.00%)
transferred 2.5 GiB of 250.0 GiB (1.00%)
transferred 5.0 GiB of 250.0 GiB (2.00%)
transferred 7.5 GiB of 250.0 GiB (3.00%)

...
<output snipped>
...

transferred 250.0 GiB of 250.0 GiB (100.00%)
unused0: successfully imported disk 'data1:100/vm-100-disk-0.raw'
```

10. Import disk 2 to the VM as a raw image. Here, import the **paragon-2.4.X-builddate-disk2.vmdk** file to the data2 datastore.

```
root@proxmox:/mnt/pve/data0/ova# qm importdisk 100 paragon-2.4.X-builddate-disk2.vmdk data2
importing disk 'paragon-2.4.X-builddate-disk2.vmdk' to VM 100 ...
Formatting '/mnt/pve/data2/images/100/vm-100-disk-0.raw', fmt=raw size=53687091200
preallocation=off
transferred 0.0 B of 50.0 GiB (0.00%)
```

```
transferred 50.0 GiB of 50.0 GiB (100.00%)
unused1: successfully imported disk 'data2:100/vm-100-disk-0.raw'
```

11. Assign the raw disks to virtio0 and virtio1 and configure the IO thread, backup, discard, and replication options and as shown.

```
root@proxmox:/mnt/pve/data0/ova# qm set 100 -virtio0 data1:100/vm-100-  
disk-0.raw,backup=0,replicate=no,discard=on,iothread=on  
update VM 100: -virtio0 data1:100/vm-100-  
disk-0.raw,backup=0,replicate=no,discard=on,iothread=on  
root@proxmox:/mnt/pve/data0/ova# qm set 100 -virtio1 data2:100/vm-100-  
disk-1.raw,backup=0,replicate=no,discard=on,iothread=on  
update VM 100: -virtio1 data2:100/vm-100-  
disk-1.raw,backup=0,replicate=no,discard=on,iothread=on
```

12. Assign a boot device for the VMs.

```
root@proxmox:/mnt/pve/data0/ova# qm set 100 --boot c --bootdisk virtio0  
update VM 100: -boot c -bootdisk virtio0
```

Disk 1 is the boot device and Disk 2 is used for Ceph storage.

13. (Optional) Configure the VM to be a non-ballooning device.

```
root@proxmox:/mnt/pve/data0/ova# qm set 100 --balloon 0  
update VM 100: -balloon 0
```

14. (Optional) If you are not using a GUI for the OS, set tablet as 0 to save on CPU and memory.

```
root@proxmox:/mnt/pve/data0/ova# qm set 100 --tablet 0  
update VM 100: -tablet 0
```

15. (Optional) Set the display option to Standard VGA to save on CPU and memory.

```
root@proxmox:/mnt/pve/data0/ova# qm set 100 --vga std  
update VM 100: -vga std
```

16. (Optional) If you want to connect to the VM console using the CLI, configure a serial terminal using a socket.

```
root@proxmox:/mnt/pve/data0/ova# qm set 100 --serial0 socket
update vm 100: -serial0 socket
```

17. Power on the VM.

```
root@proxmox:/mnt/pve/data0/ova# qm start 100
```

18. Launch the VM console.

- **Using CLI**—If you have configured a serial terminal, you can access the VM console through the CLI using the following command.

```
root@proxmox:/mnt/pve/data0/ova# qm terminal 100
```

The VM console appears.

- **Using the GUI**—Log in to the Proxmox VE GUI. Select the powered-on VM1, and click **>_ Console**. The VM console appears.

19. Repeat steps "4" on page 36 through "18" on page 39 for the other three VMs. Enter appropriate unique VM IDs and names.

You have completed the node preparation steps and created all the VMs. You are ready to configure and deploy the cluster. Go to ["Deploy the Cluster" on page 40](#).

Deploy the Cluster

SUMMARY

This topic describes the procedure to deploy a Paragon Automation cluster on the VMs.

IN THIS SECTION

- [Configure the node VMs | 40](#)
- [Deploy the cluster | 43](#)

After you have created the node VMs and prepared them, configure the cluster parameters on the VMs and deploy the Paragon Automation cluster. The steps to configure and deploy the cluster are the same regardless of the hypervisor you deploy the cluster on.

Perform the following steps to configure and deploy the Paragon Automation cluster:

1. ["Configure the node VMs" on page 40.](#)
2. ["Deploy the cluster" on page 43.](#)

Configure the node VMs

When all the node VMs are created, perform the following steps to configure the VMs.

1. Connect to the node VM Web console of the first VM. You are logged in as root automatically.
2. You are prompted to change your password immediately. Enter and re-enter the new password. You are automatically logged out of the VM.



NOTE: We recommend that you enter the same password for all the VMs. If you configure different passwords for the VMs, enter the different passwords correctly when requested to ["Generate SSH keys" on page 51](#) for the cluster nodes when deploying the cluster.

3. When prompted, log in again as root user with the newly configured password.
4. Configure the following information when prompted.

Table 3: VM Configuration Wizard

Prompt	Action
Do you want to set up a Hostname? (y/n)	Enter y to configure a hostname.
Please specify the Hostname	Enter an identifying hostname for the VM. For example: Primary1. The hostname should be under 64 characters and can include alphanumeric and some special characters. If you do not enter a hostname, a default hostname in the format controller -<VM-IP-address-4th-octet> is assigned. NOTE: Since you are deploying the cluster from one node and entering the IP addresses of the other nodes during the cluster configuration process, the roles are assigned automatically. The first three nodes to be configured are the primary and worker nodes and the last node is the worker-only node. The hostnames (and whether or not they match the role of the node), will not affect the operations of the cluster. However, for management purposes, we recommend that you pay attention to how you name the nodes and the order in which you entered their addresses during the cluster creation steps. We do not support changing the hostname after the cluster has been installed.
Do you want to set up Static IP (preferred)? (y/n)	Enter y to configure an IP address for the VM.
Please specify the IP address in CIDR notation	Enter the IP address in the CIDR notation. For example, 10.1.2.3/24. Your node VMs can be in the same subnet or in different subnets. NOTE: If you enter 10.1.2.3 instead of 10.1.2.3/24, you will see an Invalid IP address error message.

Table 3: VM Configuration Wizard (*Continued*)

Prompt	Action
Please specify the Gateway IP	Enter the gateway IP address. Ensure that you enter the gateway IP address corresponding to the network of your node VM.
Please specify the Primary DNS IP	Enter the primary DNS IP address.
Please specify the Secondary DNS IP	Enter the secondary DNS IP address.
Do you want to set up IPv6? (y/n)	Enter y to configure IPv6 addresses. If you don't want to configure IPv6 addresses, enter n and proceed to Step "5" on page 42.
Please specify the IPv6 address in CIDR notation	Enter the IPv6 address in the CIDR notation. For example, 2001:db8:1:2::3/64. Your node VMs can be in the same subnet or in different subnets. NOTE: If you enter 2001:db8:1:2::3 instead of 2001:db8:1:2::3/64, you will see an Invalid IP address error message.
Please specify the Gateway IPv6	Enter the gateway IPv6 address. Ensure that you enter the gateway IP address corresponding to the network of your node VM.
Please specify the Primary DNS IPv6	Enter the primary DNS IPv6 address.
Please specify the Secondary DNS IPv6	Enter the secondary DNS IPv6 address.

5. When prompted if you are sure to proceed, review the information displayed, type **y** and press Enter.

You are logged into Paragon Shell.

6. Repeat steps ["1" on page 40](#) through ["5" on page 42](#) for the other three VMs.

7. (Optional) Verify connectivity between the nodes. Log in again to all the node VMs. If you have been logged out, log in again as root with the previously configured ["password" on page 40](#). You are placed in Paragon Shell operational mode. Type `exit` to enter the Linux root shell of the nodes. Ping the other three nodes from each node using the `ping static-ipv4-address` command to verify that the nodes can connect to each other.
8. (Optional) Before you proceed to deploy the cluster, verify that the NTP server(s) is reachable. On any one of the cluster nodes, type `start shell`. At the `#` prompt, ping the server using the `ping ntp-servers-name-or-address` command. If the ping is unsuccessful, use an alternate NTP server.

You have completed the node preparation steps and are ready to deploy the cluster.

Deploy the cluster

Perform the following steps to configure and deploy the Paragon Automation cluster using Paragon Shell CLI.

1. Go back to the first node VM (*Primary1*). If you have been logged out, log in again as root with the previously configured ["password" on page 40](#). You are placed in Paragon Shell operational mode.

```
*****
WELCOME TO PARAGON SHELL!
You will now be able to execute Paragon CLI commands!
*****
root@eop>
```

2. To configure the cluster, enter the configuration mode in Paragon Shell.

```
root@eop> configure
Entering configuration mode

[edit]
```

3. Configure the following cluster parameters.

```
root@eop# set paragon cluster nodes kubernetes 1 address node1-IP

[edit]
root@eop# set paragon cluster nodes kubernetes 2 address node2-IP
```

```

[edit]
root@eop# set paragon cluster nodes kubernetes 3 address node3-IP

[edit]
root@eop# set paragon cluster nodes kubernetes 4 address node4-IP

[edit]
root@eop# set paragon cluster ntp ntp-servers pool.ntp.org

[edit]
root@eop# set paragon cluster common-services ingress ingress-vip generic-ingress-vIP

[edit]
root@eop# set paragon cluster applications active-assurance test-agent-gateway-vip TAGW-vIP

[edit]
root@eop# set paragon cluster applications web-ui web-admin-user "user-admin@juniper.net"

[edit]
root@eop# set paragon cluster applications web-ui web-admin-password Userpasswd

[edit]

```

Where:

The IP addresses of kubernetes nodes with indexes 1 through 4 must match the ["static IP addresses" on page 41](#) that are configured on the node VMs. The Kubernetes nodes with indexes 1,2, and 3 are the primary and worker nodes, the node with index 4 is the worker-only node. The node IP addresses can be on the same subnet or on different subnets.

ntp-servers is the NTP server to synchronize to.

web-admin-user and web-admin-password are the e-mail address and password that the first user can use to log in to the Web GUI.

ingress-vip is the VIP address for generic common ingress and is used to connect to the Web GUI.

test-agent-gateway-vip is the VIP address for the Paragon Active Assurance Test Agent gateway (TAGW).

The VIP addresses are added to the outbound SSH configuration that is required for a device to establish a connection with Paragon Automation.



NOTE: In a multi-subnet cluster installation, the VIP addresses must *not* be in the same subnet as the cluster nodes.

4. Configure the PCE server VIP address.

```
root@eop# set paragon cluster applications pathfinder pce-server pce-server-vip PCE-vIP

[edit]
```

Where:

pce-server-vip is the VIP address that is used by the PCE server to establish Path Computational Element Protocol (PCEP) sessions between Paragon Automation and the devices. The VIP address can be on the same subnet as the cluster nodes or on a different subnet. The VIP address can be on different subnets from the other VIP addresses.



NOTE: Configure the PCE server VIP address to view your network topology updates in real-time.

You can also configure the VIP address at any time post cluster deployment. For information on how to configure the PCE server VIP address after cluster deployment, see *Configure a PCE Server*.

5. Configure the routing observability feature and VIP address.

```
root@eop# set paragon cluster applications routingbot routingbot-crpd-vip v4-crpd-address

[edit]
```

Where:

routingbot-crpd-vip is the VIP address used by external network devices as BMP station IP address to establish the BMP session.



WARNING: The bare minimum resources required to configure routing observability features are listed in ["Hardware Requirements" on page 11](#). However, to get an estimate of the resources required to configure the routing observability feature on your production deployment, contact your Juniper Partner or Juniper Sales Representative.

6. (Optional) Enable the AI/ML (artificial intelligence [AI] and machine learning [ML]) feature to automatically monitor Key Performance Indicators (KPIs) related to a device's health.

```
root@eop# set paragon cluster applications aiops install-aiml true

[edit]
root@eop# set paragon cluster applications aiops enable-device-health true

[edit]
```

Where:

`install-aiml` enables AI/ML features. This is disabled, by default.

`enable-device-health` configures monitoring of device-health using AI/ML features.



WARNING: Monitoring device health using AI/ML is a Beta feature this release. The bare minimum resources required to configure the AI/ML feature is listed in ["Hardware Requirements" on page 11](#). However, to get an estimate of the resources required to configure the AI/ML feature on your production deployment, contact your Juniper Partner or Juniper Sales Representative.

7. (Optional) Configure IPv6 addresses.

```
root@eop# set paragon cluster kubernetes address-family cluster-ipv6-enabled true

[edit]
root@eop# set paragon cluster common-services ingress ingress-vip-ipv6 generic-ingress-IPv6

[edit]
root@eop# set paragon cluster applications active-assurance test-agent-gateway-vip-ipv6
TAGW-IPv6

[edit]
root@eop# set paragon cluster install prefer-ipv6 true

[edit]
```

Where:

`cluster-ipv6-enabled` enables usage of IPv6 addresses for the cluster making the cluster dual-stack.

ingress-vip-ipv6 is the IPv6 VIP address for generic common ingress and is used to connect to the Web GUI.

test-agent-gateway-vip-ipv6 is the IPv6 VIP address for the Active Assurance TAGW.

prefer-ipv6 configures preference for IPv6 addresses over IPv4 addresses. When set to true, and if hostnames are not configured, IPv6 VIP addresses are added to the outbound SSH configuration.

The VIP addresses can be on the same subnet as the cluster nodes or on a different subnet. The VIP addresses can also be on different subnets from each other.

8. (Optional) If you want to use multiple VIP addresses for generic ingress, configure the additional VIP addresses for NETCONF and gNMI.

```
root@eop# set paragon cluster common-services ingress ingress-vip octrm-gnmi-vIP

[edit]
root@eop# set paragon cluster applications oc-term oc-term-host address octrm-gnmi-vIP

[edit]
root@eop# set paragon cluster applications gnmi-term gnmi-term-host address octrm-gnmi-vIP

[edit]
```

Where:

ingress-vip is the additional VIP address to be used for NETCONF and gNMI.

oc-term-host is the VIP address that you want to use for NETCONF.

gnmi-term-host is the VIP address that you want to use for gNMI.

The address configured for NETCONF and gNMI is added to the outbound SSH configuration used to adopt devices.

9. (Optional) If your cluster nodes are in different subnets, configure BGP peering between the ToR router and the cluster nodes using the metalLB agent running in each cluster node. In this example, as illustrated in [Figure 5 on page 14](#), cluster nodes 1 and 2 are served by ToR1 and cluster nodes 3 and 4 are served by ToR2.

```
root@eop# set paragon cluster install enable-l3-vip true

[edit]
root@eop# set paragon cluster common-services metalLB metalLB-bgp-peer peer-ip ToR1-IP peer-
asn ToR1-asn local-asn node-asn local-nodes node1_IP
```

```
[edit]
root@eop# set paragon cluster common-services metallb metallb-bgp-peer peer-ip ToR1-IP peer-
asn ToR1-asn local-asn node-asn local-nodes node2_IP

[edit]
root@eop# set paragon cluster common-services metallb metallb-bgp-peer peer-ip ToR2-IP peer-
asn ToR2-asn local-asn node-asn local-nodes [node3_IP node4_IP]

[edit]
root@eop# set paragon cluster common-services metallb metallb-bgp-peer-ipv6 peer-ip ToR1-
IPv6 peer-asn ToR1-asn local-asn node-asn local-nodes [node1_IP node2_IP]

[edit]
root@eop# set paragon cluster common-services metallb metallb-bgp-peer-ipv6 peer-ip ToR2-
IPv6 peer-asn ToR2-asn local-asn node-asn local-nodes [node3_IP node4_IP]

[edit]
```

Where:

`enable-l3-vip` enables L3 VIP addresses for cluster nodes and VIP addresses in different subnets.

`metallb-bgp-peer` and `metallb-bgp-peer-ipv6` are the IP and IPv6 addresses of the ToR routers, respectively.

`peer-asn` is the ToR AS number.

`local-asn` is the AS number of the cluster nodes. The AS number remains the same for all the cluster nodes.

`local-nodes` are the cluster nodes IP addresses configured in step "3" on page 43.

10. (Optional) If you want to configure hostnames for generic ingress and Paragon Active Assurance TAGW, configure the following:

```
root@eop# set paragon cluster common-services ingress system-hostname ingress-vip-dns-
hostname

[edit]
root@eop# set paragon cluster applications active-assurance test-agent-gateway-hostname
nginx-ingress-controller-hostname

[edit]
```

Where:

system-hostname is the hostname for the generic ingress virtual IP (VIP) address.

test-agent-gateway-hostname is the hostname for the Paragon Active Assurance TAGW VIP address.

When you configure hostnames, the hostnames take precedence over VIP addresses and are added to the outbound SSH configuration. The hostnames can resolve to either IPv4 or IPv6 VIP addresses or both.

11. (Optional) Configure the following settings for SMTP-based user management.

```

root@eop# set paragon cluster mail-server smtp-relayhost smtp.relayhost.com

[edit]
root@eop# set paragon cluster mail-server smtp-relayhost-username relayuser

[edit]

root@eop# set paragon cluster mail-server smtp-relayhost-password relaypassword
[edit]

root@eop# set paragon cluster mail-server smtp-allowed-sender-domains paragonautomation.net

[edit]
root@eop# set paragon cluster mail-server smtp-sender-email no-reply@paragonautomation.net

[edit]
root@eop# set paragon cluster mail-server smtp-sender-name Juniper Paragon Automation

[edit]
root@eop# set paragon cluster papi papi-local-user-management false

[edit]
root@eop# set paragon cluster mail-server smtp-enabled true

[edit]

```

Where:

smtp-allowed-sender-domains are the e-mail domains from which Paragon Automation sends e-mails to users.

smtp-relayhost is the name of the SMTP server that relays messages.

smtp-relayhost-username (optional) is the username to access the SMTP (relay) server.

smtp-relayhost-password (optional) is the password for the SMTP (relay) server.

smtp-allowed-sender-domains are the e-mail domains from which Paragon Automation sends e-mails to users.

smtp-sender-email is the e-mail address that appears as the sender's e-mail address to the e-mail recipient.

smtp-sender-name is the name that appears as the sender's name in the e-mails sent to users from Paragon Automation.

papi-local-user-management enables or disables local user management.

mail-server smtp-enabled enables or disables SMTP.



NOTE: SMTP configuration is optional at this point. SMTP settings can be configured after the cluster has been deployed also. For information on how to configure SMTP after cluster deployment, see *Configure SMTP Settings in Paragon Shell*.

12. (Optional) Install custom user certificates. Note, before you install user certificates, you must copy the custom certificate file and certificate key file to the Linux root shell of the node from which you are deploying the cluster. Copy the files to the **/root/epic/config** folder.

```
root@eop# set paragon cluster common-services ingress user-certificate use-user-certificate
true

[edit]
root@eop# set paragon cluster common-services ingress user-certificate user-certificate-
filename "certificate.cert.pem"

[edit]
root@eop# set paragon cluster common-services ingress user-certificate user-certificate-key-
filename "certificate.key.pem"

[edit]
```

Where:

user-certificate-filename is the user certificate filename.

user-certificate-key-filename is the user certificate key filename.



NOTE: Installing certificates is optional at this point. You can configure Paragon Automation to use custom user certificates after cluster deployment also. For information on how to install user certificates after cluster deployment, see ["Install User Certificates" on page 57](#).

13. Commit the configuration and exit configuration mode.

```
root@eop# commit
commit complete

[edit]
root@eop# exit
Exiting configuration mode

root@eop>
```

14. Generate the configuration files.

```
root@eop> request paragon config
Paragon inventory file saved at /epic/config/inventory
Paragon config file saved at /epic/config/config.yml
```

The **inventory** file contains the IP addresses of the VMs.

The **config.yml** file contains minimum Paragon Automation cluster configuration parameters that are required to deploy a cluster.

The request paragon config command also generates a **config.cmgd** file in the **config** directory. The **config.cmgd** file contains all the set commands that you have executed. If the **config.yml** file is inadvertently edited or corrupted, you can redeploy your cluster using the load set config/config.cmgd command in the configuration mode.

15. Generate SSH keys on the cluster nodes.

When prompted, enter the SSH password for the VMs. Enter the same ["password" on page 40](#) that you configured to log in to the VMs.

```
root@eop> request paragon ssh-key
Setting up public key authentication for ['node1-IP', 'node2-IP', 'node3-IP', 'node4-IP']
```

```

Please enter SSH username for the node(s): root
Please enter SSH password for the node(s): password
checking server reachability and ssh connectivity ...
Connectivity ok for node1-IP
Connectivity ok for node2-IP
Connectivity ok for node3-IP
Connectivity ok for node4-IP
SSH key pair generated in node1-IP
SSH key pair generated in node2-IP
SSH key pair generated in node3-IP
SSH key pair generated in node4-IP
copied from node1-IP to node1-IP
copied from node1-IP to node2-IP
copied from node1-IP to node3-IP
copied from node1-IP to node4-IP
copied from node2-IP to node1-IP
copied from node2-IP to node2-IP
copied from node2-IP to node3-IP
copied from node2-IP to node4-IP
copied from node3-IP to node1-IP
copied from node3-IP to node2-IP
copied from node3-IP to node3-IP
copied from node3-IP to node4-IP
copied from node4-IP to node1-IP
copied from node4-IP to node2-IP
copied from node4-IP to node3-IP
copied from node4-IP to node4-IP

```



NOTE: If you have configured different passwords for the VMs, ensure that you enter corresponding passwords when prompted.

16. Deploy the cluster.

```

root@eop> request paragon deploy cluster
Process running with PID: 231xx03
To track progress, run 'monitor start /epic/config/log'
After successful deployment, please exit Paragon-shell and then re-login to the host to
finalize the setup

```

The cluster deployment begins and takes over an hour to complete.



NOTE: When you run the request `paragon deploy cluster` command, sometimes the command may fail because the **config.yml** is empty. Verify that the **config.yml** is empty using the file `show /epic/config/config.yml` command. Perform the steps detailed in the [Release Notes: Installation and Upgrade](#) section to repopulate the **config.yml** file and reattempt deploying the cluster.

17. (Optional) Monitor the progress of the deployment onscreen.

```
root@eop> monitor start /epic/config/log
```

The progress of the deployment is displayed. Deployment is complete when you see an output similar to this onscreen.

```
<output snipped>
PLAY RECAP *****
node1-IP      : ok=2397 changed=1015 unreachable=0    failed=0    rescued=0
ignored=15
node2-IP      : ok=192  changed=96  unreachable=0    failed=0    rescued=0
ignored=0
node3-IP      : ok=192  changed=96  unreachable=0    failed=0    rescued=0
ignored=0
node4-IP      : ok=186  changed=95  unreachable=0    failed=0    rescued=0
ignored=0

Thursday 27 February 2025  22:00:57 +0000 (0:00:00.183)      1:01:53.469 *****
=====
user-registry : Push Docker Images from local registry to paragon registry - 335.76s
kubernetes/addons/rook : Wait for Object-Store ----- 213.36s
kubernetes/multi-master-rke2 : start rke2 server on 1st master ----- 212.18s
jcloud/papi : wait for papi rest api ----- 122.19s
jcloud/airflow2 : Install Helm Chart ----- 117.00s
Check if kafka container is up ----- 108.86s
Install Helm Chart ----- 106.30s
kubernetes/addons/postgres-operator : Make sure postgres is fully up and accepting request
using regular user -- 73.73s
systemd ----- 62.14s
kubernetes/addons/postgres-operator-rb : RB >>> Make sure postgres is fully up and
accepting request using regular user -- 52.96s
kubernetes/multi-master-rke2 : start rke2 server on other master ----- 51.71s
Create Kafka Topics ----- 51.48s
```

```

systemcheck : Get Disk IOPS ----- 49.09s
delete existing install config-map - if any ----- 43.10s
paa/timescaledb : Make sure postgres is fully up and accepting request using regular user
-- 41.96s
kubernetes/multi-master-rke2 : start rke2 server on other master ----- 41.90s
Save installer config to configmap ----- 41.47s
Install Helm Chart ----- 35.92s
Verify healthbot victoriametrics ----- 35.17s
kubernetes/addons/vm-operator-rb : Wait for vm storage statefulset and pods -- 31.15s
Playbook run took 0 days, 1 hours, 1 minutes, 53 seconds

root@eop>

```

Alternatively, if you did not choose to monitor the progress of the deployment onscreen using the `monitor` command, you can view the contents of the log file using the `file show /epic/config/log` command. The last few lines of the log file must look similar to the ["sample output" on page 53](#). We recommend that you check the log file periodically to monitor the progress of the deployment.

18. Upon successful completion of the deployment, the application cluster is created. Log out of the VM and log in again to Paragon Shell.

The console output displays the Paragon Shell welcome message and the IP addresses of the four nodes (called Controller-1 through Controller-4), the Paragon Active Assurance TAGW VIP address, the Web admin user e-mail address, and Web GUI IP address. If IPv6 addresses are configured, the welcome message displays the IPv6 VIP addresses as well.

```

Welcome to Juniper Paragon Automation OVA

This Controller IP: node1-IP, node1-IPv6
This VM node1-IP, node1-IPv6 is part of an EPIC on-prem system with IPv6 enabled.
=====
Controller IP      : node1-IP, node2-IP, node3-IP, node4-IP
PAA Virtual IP    : TAGW-vIP, TAGW-vIPv6
UI                : https://generic-ingress-vIP, https://[generic-ingress-vIPv6]
Web Admin User    : admin-user@juniper.net
=====
ova: 20250526_1117
build: eop-2.4.X.8952.gbef82aec6b

*****
WELCOME TO PARAGON SHELL!
You will now be able to execute Paragon CLI commands!

```

```
*****
root@Primary1>
```

The CLI command prompt displays your login username and the node hostname that you configured previously. For example, if you entered Primary1 as the hostname of your primary node, the command prompt is root@Primary1 >.

You can now verify the cluster deployment and log in to the Web GUI. If you are accessing the Web GUI from an external IP address, outside the Paragon Automation network, you must use NAT to map the external IP address to the Web GUI IP address. Go to ["Log in to the Web GUI" on page 55](#).

Log in to the Web GUI

After the cluster has been deployed, you can verify the deployment (optionally), and log in to the Web GUI using the information you entered during the deployment.

1. (Optional) Verify the deployment.

a. Verify cluster details.

```
root@Primary1 > show paragon cluster details
Storage and Controller Node IPs: node1-IP, node2-IP, node3-IP, node4-IP
```

b. Verify cluster node details.

```
root@Primary1 > show paragon cluster nodes
```

NAME	STATUS	ROLES	AGE	VERSION
Primary1	Ready	control-plane,etcd,master	4h12m	v1.31.1+rke2r1
Primary2	Ready	control-plane,etcd,master	4h12m	v1.31.1+rke2r1
Primary3	Ready	control-plane,etcd,master	4h11m	v1.31.1+rke2r1
Worker1	Ready	influxdb-worker,worker	4h11m	v1.31.1+rke2r1

2. Log in to the Web GUI.

- Enter the common ingress VIP address in a Web browser to access the Paragon Automation login page. The common ingress IP address, that you configured during installation, can be either IPv4 or IPv6. The URLs to the UI is displayed on your console welcome message after the cluster deployment is complete.

To use the IPv4 address to connect to the Web GUI, enter **`https://generic-ingress-vip`** in the URL. For example, **`https://10.1.2.7`**.

To use the IPv6 address to connect to the Web GUI, enter **`https://[generic-ingress-vipv6]`** in the URL. Ensure that you enclose the IPv6 address within square brackets. For example, **`https://[2001:db8:1:2::7]`**.

Alternatively, if you have configured hostnames, enter **`https://ingress-vip-dns-hostname`** to connect to the Web GUI.

- b. Enter the Web admin user e-mail and password that you configured previously to log in to Paragon Automation. The Web admin user e-mail is also displayed on your console after the cluster deployment is complete.

You are logged in to the Paragon Automation GUI and are directed to the New Account page from where you can create a new Organization. For more information, see *User Activation and Login*.

For a list of cluster-related tasks that you can perform using Paragon Shell post installation of the Paragon Automation cluster, see ["Post Installation Tasks" on page 56](#).

Post Installation Tasks

IN THIS SECTION

- [Install User Certificates | 57](#)
- [Update Victoria Metrics | 59](#)

After you install the Paragon Automation cluster, you can use Paragon Shell to perform the following additional optional tasks:

- Ensure that the installed cluster is healthy and operational.

Execute the request `paragon health-check` command. The Overall Cluster Status must be GREEN. For example:

```
root@primary1> request paragon health-check
Health status checking...
```

```

=====
Get node count of Kubernetes cluster.
=====

OK
There are 4 nodes in the cluster.
...
<output snipped>
...
=====

Verifying Elasticsearch
=====

OK
Opensearch test...
Checking health status at opensearch-cluster-master.common:9200...
Opensearch is healthy (green).
OPENSEARCH VERIFICATION PASS

=====

Overall cluster status
=====

GREEN

```

- Configure the PCE server. See *Configure PCE Server*.
- Configure SMTP-based user management. See *Configure SMTP Settings in Paragon Shell*.
- Install custom user certificates. See ["Install User Certificates" on page 57](#).
- Back up your Paragon Automation cluster configuration. See ["Back Up Using Paragon Shell" on page 103](#).
- Configure monitoring in Paragon Automation to collect metrics from different types of sources and forward the collected data to designated sinks. See [Configure Monitoring](#).
- Update the Victoria Metrics cluster configuration parameters. See ["Update Victoria Metrics" on page 59](#).

Install User Certificates

You can configure Paragon Automation to use custom user certificates. You can install user certificates either when you deploy the Paragon Automation cluster or post deployment.

To upload and install custom user certificates, perform the following steps:

1. Log in to the Linux root shell of the node from which you deployed the cluster.
2. Copy the custom user certificate file and key file to the **root/epic/config** directory.
3. Type `cli` to log in to Paragon Shell and type `configure` to enter configuration mode.
4. Configure the following parameters, commit, and exit the configuration mode.

```
root@primary1# set paragon cluster common-services ingress user-certificate use-user-
certificate true

root@primary1# set paragon cluster common-services ingress user-certificate user-certificate-
filename "certificate.cert.pem"

root@primary1# set paragon cluster common-services ingress user-certificate user-certificate-
key-filename "certificate.key.pem"

root@primary1# commit
commit complete

[edit]
root@primary1# exit
Exiting configuration mode

root@primary1>
```

Where:

certificate.cert.pem is the user certificate filename.

certificate.key.pem is the user certificate key filename.

5. Regenerate the configuration files.

```
root@primary1> request paragon config
Paragon inventory file saved at /epic/config/inventory
Paragon config file saved at /epic/config/config.yml
```

6. Deploy the certificates.

```
root@primary1> request paragon deploy cluster input "-t ingress-controller"
Process running with PID: 23xx022
To track progress, run 'monitor start /epic/config/log'
```

Update Victoria Metrics

Update the Victoria Metrics cluster configuration parameters. Victoria-metrics is a time series database. Different microservices populate the database and query data using the APIs provided by Victoria metrics.

1. Log in to the node from which you deployed the Paragon Automation cluster.
2. Type `configure` to enter configuration mode.
3. Update the Victoria Metrics parameters. Use the following commands.

- ```
root@primary1# set paragon cluster insights victoria-metrics vm-cluster-replication-factor
replication_factor
```
- ```
root@primary1# set paragon cluster insights victoria-metrics vm-cluster-retention-period
retention_period
```
- ```
root@primary1# set paragon cluster insights victoria-metrics vm-cluster-storage-instances
number_of_pods
```

For more information on `vm-cluster-replication-factor`, `vm-cluster-retention-period`, and `vm-cluster-storage-instances`, see *set paragon cluster insights victoria-metrics*.

4. Type `commit` and `quit` to commit the configuration and exit configuration mode.
5. Regenerate the configuration files.

```
root@primary1> request paragon config
Paragon inventory file saved at /epic/config/inventory
Paragon config file saved at /epic/config/config.yml
```

6. Deploy the updated configuration.

```
root@primary1> request paragon deploy cluster input "-v -t victoriametrics-cluster"
Process running with PID: 23xx022
To track progress, run 'monitor start /epic/config/log'
```

## RELATED DOCUMENTATION

*Configure a PCE Server*

*Configure SMTP Settings in Paragon Shell*

# 4

CHAPTER

## Upgrade the Cluster

---

### IN THIS CHAPTER

- Upgrade Paragon Automation | 62
  - Preupgrade Paragon Shell Before Upgrade | 72
-

# Upgrade Paragon Automation

## IN THIS SECTION

- Upgrade Prerequisites | 63
- Upgrade the Paragon Automation cluster | 64
- Upgrade Paragon Shell and the OVA System Files | 70
- Post Cluster Upgrade Tasks | 71

The upgrade functionality provided by Paragon Shell enables you to upgrade your Paragon Automation installation and all the applications running on it to the current release.

You can upgrade to the current Juniper Paragon Automation Release 2.4.X from the following releases.

- Release 2.3.0
- Release 2.2.0

We do not support upgrading directly from Juniper Paragon Automation releases 2.0.0 and 2.1.0 to release 2.4.X. If you have a release 2.1.0 installation, you can upgrade to release 2.2.0, and then use the process described in this topic to subsequently upgrade to release 2.4.X.

The upgrade process is automated by a set of Paragon Shell commands and carries out the required system checks, retrieves the upgrade package, and executes the upgrade process on the cluster nodes. You can upgrade using a file that is either downloaded locally on your primary node or downloaded directly from a Web page.

During an upgrade, it is important that no change activities including onboarding of devices, provisioning of services or changing other configurations are done in the system. The upgrade will automatically reboot all components and there will be short unavailability during that time. The upgrade process does not affect the traffic through the network and once the upgrade is complete, the devices and services are not reconfigured.



**NOTE:** We recommend that you back up your configuration before upgrading. For information on backing up your current configuration, see ["Back Up and Restore Paragon Automation" on page 103](#).

Perform the following steps to upgrade to Paragon Automation Release 2.4.X:

1. ["Upgrade Prerequisites" on page 63](#)—Ensure that all upgrade prerequisites are met
2. ["Upgrade the Paragon Automation cluster" on page 64](#)—Upgrade the cluster using either ["Upgrade using the local filename Option" on page 65](#) or ["Upgrade using the remote url Option" on page 67](#)
3. ["Upgrade Paragon Shell and the OVA System Files" on page 70](#)—Upgrade Paragon Shell and the OVA system files on all the cluster nodes

## Upgrade Prerequisites

Before you upgrade the Paragon Automation cluster, ensure the following.

- Paragon Shell is accessible and operational.
- The cluster nodes have the following free disk space available:
  - The primary node from which the cluster was deployed must have 15% of the total disk space + three times the upgrade file size free.
  - The other two primary and worker nodes must have 15% of the total disk space + the same amount as the upgrade file size free.
  - The worker node must have 15% of the total disk space free.
- Disable and delete previous OpenSearch backup files to free up space.
  1. Disable OpenSearch backup.

```
root@primary1# kubectl patch cronjob opensearch-backup-cron -n common -p '{"spec": {"suspend": true}}'
```

2. Delete the periodic backup job.

```
root@primary1# kubectl delete job -n common -l app=opensearch-backup-cron
```

3. Delete all existing OpenSearch backup files.

```
root@primary1# kubectl exec -i -n common -c opensearch-backup $(kubectl get po -n common -l app=opensearch-backup -o jsonpath={.items[0].metadata.name}) -- bash -c 'rm -rf /opt/paragon/opensearch-backup/*'
```

- Verify that the cluster is healthy and operational.

If you are upgrading from release 2.2.0 to release 2.4.X, execute the health-check command from the Linux root shell. The Overall Cluster Status must be GREEN.

```
root@primary1# health-check
Health status checking...

=====
Get node count of Kubernetes cluster.
=====

OK
There are 4 nodes in the cluster.
...
<output snipped>
...

=====
Overall cluster status
=====

GREEN
```

If you are upgrading from release 2.3.0 to release 2.4.X, the upgrade command in release 2.3.0 preemptively runs the health-check to verify cluster health before upgrading the cluster. However, we recommend that you check cluster health before using the upgrade command. Execute the request `paragon health-check` command from Paragon Shell. The Overall Cluster Status must be GREEN.

- (Optional) Check the current build and OVA version of your existing release from Paragon Shell using the `show paragon version` command.

## Upgrade the Paragon Automation cluster

### IN THIS SECTION

- [Upgrade using the local filename Option | 65](#)
- [Upgrade using the remote url Option | 67](#)

Perform the following steps if you want to upgrade Paragon Automaton releases 2.3.0 and 2.2.0 to the current 2.4.X release.

You can upgrade your installation and all the applications running on it using *any one* of the following two options:

## Upgrade using the local filename Option

Use this option for air-gapped environments where your Paragon Automation installation does not have access to the Internet. However, you need to be able to copy the **upgrade\_paragon-release-build-id.tgz** and **upgrade\_paragon-release-build-id.tgz.psig** files to your primary node.



**NOTE:** If you are upgrading from release 2.2.0 to release 2.4.X, perform the steps detailed in [Upgrade using the local Option](#).

If you are upgrading from release 2.3.0 to release 2.4.X, perform the following steps.

1. Log in as root user to the primary node from which the current cluster was installed. You are logged in to Paragon Shell.
2. Type `exit` to exit from Paragon Shell to the Linux root shell.
3. Copy the **upgrade\_paragon-release-build-id.tgz** and **upgrade\_paragon-release-build-id.tgz.psig** files, of the version to which you want to upgrade, to the `/root/epic/temp` folder.

You might need to download the **upgrade\_paragon-release-build-id.tgz** and **upgrade\_paragon-release-build-id.tgz.psig** files from the *Juniper Software Download* site to your local computer before copying it to the primary node.

4. (Optional) Use the `gpg --verify` command to validate the digital signature of the upgrade file. For example:

```
root@primary1:~/epic/temp# gpg --verify upgrade_paragon-
release-2.4.X.8952.gbef82aec6b.tgz.psig upgrade_paragon-release-2.4.X.8952.gbef82aec6b.tgz
gpg: Signature made Sat Mar 01 01:00:09 2024 UTC
gpg: using RSA key 4B7B22C9C4FE32CF
gpg: Good signature from "Northstar Paragon Automation 2024 ca@juniper.net" [ultimate]
```

Here `primary1` is the installer primary node. Validation takes a couple of minutes to complete.

5. Type `cli` to enter Paragon Shell.
6. Use the following command to upgrade the Paragon Automation cluster:

```
request paragon cluster upgrade local filename upgrade_paragon-release-build-id.tgz
```

For example:

```
root@primary1> request paragon cluster upgrade local filename upgrade_paragon-
release-2.4.X.8952.gbef82aec6b.tgz
Upgrade is in progress ...
Updated to build: paragon-release-2.4.X.8952.gbef82aec6b
Paragon Cluster upgrade is successful!
Run 'request paragon health-check' command to check current system health with upgraded
Paragon cluster.
Please continue to primary host node to upgrade Paragon-shell and update OVA system files by:
/root/epic/upgrade_paragon-shell_ova-system.sh
```

Here primary1 is the installer primary node. The upgrade command checks the health of the cluster before upgrading. If the cluster health check returns a GREEN status, the cluster is upgraded requiring no further input. If the cluster health check returns a RED status, the cluster is not upgraded. If the cluster health check returns an AMBER status, you are prompted to choose to continue or stop the upgrade.

You can also use the request paragon cluster upgrade local filename upgrade\_paragon-release-*build-id*.tgz no-confirm option to ignore the AMBER status and continue with the upgrade without being prompted. The no-confirm option does not ignore a RED status.

Note that, the upgrade process takes over an hour to complete. If you get disconnected from the VM during the upgrade process, you can periodically check the upgrade log file until you see an output similar to this:

```
root@primary1:~# cat /root/upgrade/upgrade.log
<output snipped>
...
PLAY RECAP *****
10.1.2.3 : ok=1819 changed=430 unreachable=0 failed=0 rescued=0
ignored=2
10.1.2.4 : ok=185 changed=26 unreachable=0 failed=0 rescued=0
ignored=0
10.1.2.5 : ok=185 changed=26 unreachable=0 failed=0 rescued=0
ignored=0
10.1.2.6 : ok=177 changed=25 unreachable=0 failed=0 rescued=0
ignored=0

Saturday 01 March 2025 09:41:53 +0000 (0:00:00.665) 1:26:57.926 *****
=====
user-registry : Push Docker Images from local registry to paragon registry - 532.34s
```

```

jcloud/airflow2 : Install Helm Chart ----- 278.28s
Install Helm Chart ----- 147.88s
delete existing install config-map - if any ----- 111.87s
Save installer config to configmap ----- 98.15s
jcloud/papi : Install Helm Chart ----- 97.77s
Create Kafka Topics ----- 79.97s
user-registry : Push Helm Charts to paragon registry ----- 78.70s
systemd ----- 67.23s
kubernetes/addons/helper-commands : Install Pathfinder Utility scripts -- 44.65s
kubernetes/addons/helper-commands : Copy profiler to /opt/paragon/bin -- 39.79s
registry : Copy nginx image on 10.1.2.4 ----- 37.46s
registry : Copy nginx image on 10.1.2.5 ----- 37.04s
registry : Copy nginx image on 10.1.2.6 ----- 36.80s
registry : Copy nginx image on 10.1.2.3 ----- 36.03s
Install Helm Chart ----- 34.49s
registry : Copy zot image on 10.1.2.4 ----- 33.29s
registry : Copy zot image on 10.1.2.5 ----- 32.46s
registry : Copy zot image on 10.1.2.6 ----- 31.67s
registry : Copy zot image on 10.1.2.3 ----- 30.25s
Playbook run took 0 days, 1 hours, 26 minutes, 57 seconds
registry-14272
Application Cluster upgraded to version build: paragon-release-2.4.X.8952.gbef82aec6b!!!

```

Your Paragon Automation installation and all the applications running on it are upgraded.

7. Execute the request `paragon health-check` command to ensure that the upgraded cluster is healthy and operational.

The Overall Cluster Status must be GREEN.

8. Upgrade Paragon Shell and the OVA system files.

Go to ["Upgrade Paragon Shell and the OVA System Files" on page 70](#).

## Upgrade using the remote url Option

Use this option if your Paragon Automation installation has access to the Internet and the upgrade file is in a remote location.



**NOTE:** If you are upgrading from release 2.2.0 to release 2.4.X, perform the steps detailed in [Upgrade using the url Option](#). The remote url option is available only from release 2.3.0 onwards.

If you are upgrading from release 2.3.0 to release 2.4.X, perform the following steps.

1. Log in as root user to the primary node from which your existing cluster was installed. You are logged in to Paragon Shell.
2. Use the following command to upgrade the Paragon Automation cluster:

```
request paragon cluster upgrade remote url "https://juniper.software.download.site/upgrade_paragon-release-build-id.tgz?query_string"
```

For example:

```
root@primary1> request paragon cluster upgrade remote url "https://cdn.juniper.net/software/
paragon-images/upgrade_paragon-release-2.4.X.8952.gbef82aec6b.tgz?query_string"
Checking paragon cluster system health before proceeding with cluster upgrade. This will take
a minute...
...
<output snipped>
...
=====
Overall cluster status
=====

GREEN

=====

Paragon cluster is healthy.
Proceed with Paragon cluster upgrade.
Upgrading paragon cluster from https://cdn.juniper.net/software/paragon-images
Downloading tarball file upgrade_paragon-release-2.4.X.8952.gbef82aec6b
Download file size: 28,831,677,064 bytes
Current disk Usage:
 Total: 263,622,004,736 bytes
 Used: 106,109,399,040 bytes
 Available: 145,685,159,936 bytes
Please wait for current download to finish... (File is large. It may take a while.)
Upgrade tarball file is downloaded.
Upgrade is in progress ...
Updated to build: eop-release-2.4.X.8952.gbef82aec6b
Paragon Cluster upgrade is successful!
Run 'request paragon health-check' command to check current system health with upgraded
Paragon cluster.
Please continue to primary host node to upgrade Paragon-shell and update OVA system files by:
/root/epic/upgrade_paragon-shell_ova-system.sh
```

Here primary1 is the installer primary node. The upgrade command checks the health of the cluster before upgrading. If the cluster health check returns a GREEN status, the cluster is upgraded requiring no further input. If the cluster health check returns a RED status, the cluster is not upgraded. If the cluster health check returns an AMBER status, you are prompted to choose to continue or stop the upgrade.

You can also use the request paragon cluster **upgrade remote url** "[https://juniper.software.download.site/upgrade\\_paragon-release-build-id.tgz?query\\_string](https://juniper.software.download.site/upgrade_paragon-release-build-id.tgz?query_string)" no-confirm option to ignore the AMBER status and continue with the upgrade without being prompted. The no-confirm option does not ignore a RED status.

Note that, the upgrade process takes a little over an hour to complete. If you get disconnected from the VM during the upgrade process, you can periodically check the upgrade log file until you see an output similar to this:

```
root@primary1:~# cat /root/upgrade/upgrade.log
<output snipped>
...
PLAY RECAP *****
10.1.2.3 : ok=1819 changed=430 unreachable=0 failed=0 rescued=0
ignored=2
10.1.2.4 : ok=185 changed=26 unreachable=0 failed=0 rescued=0
ignored=0
10.1.2.5 : ok=185 changed=26 unreachable=0 failed=0 rescued=0
ignored=0
10.1.2.6 : ok=177 changed=25 unreachable=0 failed=0 rescued=0
ignored=0

Saturday 01 March 2025 09:41:53 +0000 (0:00:00.665) 1:26:57.926 *****
=====
user-registry : Push Docker Images from local registry to paragon registry - 532.34s
jcloud/airflow2 : Install Helm Chart ----- 278.28s
Install Helm Chart ----- 147.88s
delete existing install config-map - if any ----- 111.87s
Save installer config to configmap ----- 98.15s
jcloud/papi : Install Helm Chart ----- 97.77s
Create Kafka Topics ----- 79.97s
user-registry : Push Helm Charts to paragon registry ----- 78.70s
systemd ----- 67.23s
kubernetes/addons/helper-commands : Install Pathfinder Utility scripts -- 44.65s
kubernetes/addons/helper-commands : Copy profiler to /opt/paragon/bin -- 39.79s
registry : Copy nginx image on 10.1.2.4 ----- 37.46s
registry : Copy nginx image on 10.1.2.5 ----- 37.04s
```

```

registry : Copy nginx image on 10.1.2.6 ----- 36.80s
registry : Copy nginx image on 10.1.2.3 ----- 36.03s
Install Helm Chart ----- 34.49s
registry : Copy zot image on 10.1.2.4 ----- 33.29s
registry : Copy zot image on 10.1.2.5 ----- 32.46s
registry : Copy zot image on 10.1.2.6 ----- 31.67s
registry : Copy zot image on 10.1.2.3 ----- 30.25s
Playbook run took 0 days, 1 hours, 26 minutes, 57 seconds
registry-14272
Application Cluster upgraded to version build: paragon-release-2.4.X.8952.gbef82aec6b!!!

```

Your Paragon Automation installation and all the applications running on it are upgraded.

3. Execute the request `paragon health-check` command to ensure that the upgraded cluster is healthy and operational.

The Overall Cluster Status must be GREEN.

4. Upgrade Paragon Shell and the OVA system files.

Go to ["Upgrade Paragon Shell and the OVA System Files" on page 70.](#)

## Upgrade Paragon Shell and the OVA System Files

When your Paragon Automation installation and all the applications running on it are successfully upgraded, you must upgrade Paragon Shell and the OVA system files.

1. Exit from the installer primary node Paragon Shell to the Linux root shell by typing `exit`.
2. Execute the Paragon Shell upgrade shell script.

```

root@primary1:~# bash /root/epic/upgrade_paragon-shell_ova-system.sh
Upgrading paragon-shell...
Updating paragon-shell for primary1.....
Container paragon-shell Stopping
Container paragon-shell Stopped
Container paragon-shell Removing
Container paragon-shell Removed
paragon-shell Pulling
....
<output snipped>
....

```

```

primaryname update-status
primary1 ok
primary3 ok
primary2 ok
primary4 ok
paragon-shell upgrade successful!
Updating OVA system files...
OVA system files update successful!

```

Paragon Shell and the OVA system files are upgraded.

3. (Optional) Check the build and OVA version of your upgraded cluster from Paragon Shell.

```

root@primary> show paragon version
ova: 20250301_1117_ova
ova-patch: 20250301_0349
build: eop-2.4.X.8952.gbef82aec6b
Client Version: v1.29.6
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Server Version: v1.31.4+rke2r1

```

Proceed to perform the post cluster upgrade tasks.

## Post Cluster Upgrade Tasks

After upgrading the cluster and Paragon Shell OVA, perform the following tasks to complete the upgrade process.

1. Update the base OS. See ["Update the OS" on page 85](#).
2. Upgrade the service designs, update the network implementation plan, and recreate the resource and service instances. See *Update the Network Implementation Plan and Recreate Service Instances After Upgrade*.

### RELATED DOCUMENTATION

[Update the OS | 85](#)

*Update the Network Implementation Plan and Recreate Service Instances After Upgrade*

[Back Up and Restore Paragon Automation | 103](#)

# Preupgrade Paragon Shell Before Upgrade

## SUMMARY

Use this procedure only if you are an advanced user, or have upgraded an earlier release of your cluster successfully before, or have been directed by a Juniper Partner to preupgrade.

## IN THIS SECTION

- [Upgrade Prerequisites | 73](#)
- [Perform Preupgrade | 74](#)
- [Upgrade the Paragon Automation Cluster | 75](#)
- [Upgrade Paragon Shell and the OVA System Files | 82](#)
- [Post Cluster Upgrade Tasks | 83](#)

The upgrade command implicitly backs up the OpenSearch database. In releases earlier than release 2.4.X, the upgrade command backed up the whole OpenSearch database. If the data volume in OpenSearch is large, the upgrade becomes a time intensive process. In release 2.4.X, the upgrade command backs-up only critical OpenSearch data, thereby reducing the time taken to upgrade. However, to use the upgrade command supported in release 2.4.X, you must execute a preupgrade script to upgrade Paragon Shell on the installer primary node of your existing earlier release before you upgrade the release. You can then use the upgrade commands options supported in release 2.4.X to perform the upgrade. Additionally, if you want to separately back up all OpenSearch data and not just critical data, the preupgrade script also enables you to manually take a back up of all the data before upgrading the cluster.

Perform the following steps to upgrade the Paragon Automation Release 2.2.0 or release 2.3.0 to release 2.4.X after using the preupgrade script:

1. ["Upgrade Prerequisites" on page 73](#)—Ensure that all upgrade prerequisites are met
2. ["Perform Preupgrade" on page 74](#)—Use the preupgrade script to upgrade Paragon Shell on the installer primary node and manually back up OpenSearch data
3. ["Upgrade the Paragon Automation Cluster" on page 75](#)—Upgrade the cluster using either ["Upgrade using the local Option" on page 75](#) or ["Upgrade using the remote url Option" on page 78](#)
4. ["Upgrade Paragon Shell and the OVA System Files" on page 82](#)—Upgrade Paragon Shell and the OVA system files on all the cluster nodes
5. ["Post Cluster Upgrade Tasks" on page 83](#)—Update the base OS and recreate service and resources instances.

## Upgrade Prerequisites

Before you upgrade the Paragon Automation cluster, ensure the following.

- Paragon Shell is accessible and operational.
- The cluster nodes have the following free disk space available:
  - The primary node from which the cluster was deployed must have 15% of the total disk space + three times the upgrade file size free.
  - The other two primary and worker nodes must have 15% of the total disk space + the same amount as the upgrade file size free.
  - The worker node must have 15% of the total disk space free.
- Disable and delete previous OpenSearch backup files to free up space.

### 1. Disable OpenSearch backup.

```
root@primary1# kubectl patch cronjob opensearch-backup-cron -n common -p '{"spec": {"suspend": true}}'
```

### 2. Delete the periodic backup job.

```
root@primary1# kubectl delete job -n common -l app=opensearch-backup-cron
```

### 3. Delete all existing OpenSearch backup files.

```
root@primary1# kubectl exec -i -n common -c opensearch-backup $(kubectl get po -n common -l app=opensearch-backup -o jsonpath={.items[0].metadata.name}) -- bash -c 'rm -rf /opt/paragon/opensearch-backup/*'
```

- (Optional) Check the current build and OVA version of your existing setup from Paragon Shell using the `show paragon version` command.

Go to ["Perform Preupgrade" on page 74](#) to preupgrade the cluster.

## Perform Preupgrade

Perform the following steps to download the preupgrade script file and prepare the cluster for upgrade.

1. Log in as root user to the primary node from which the current cluster was installed. You are logged in to Paragon Shell.
2. Type `exit` to exit from Paragon Shell to the Linux root shell.
3. Copy the **preupgrade.sh** file to the **/root/epic/temp** folder or any location in your primary node.

In air-gapped environments where your Paragon Automation installation does not have access to the Internet, you must ensure you are able to securely copy the script file to the primary node. You might need to download the **preupgrade.sh** file from the *Juniper Software Download* site to your local computer before copying it to the primary node.

4. Navigate to the download location and make the **preupgrade.sh** executable (if not already executable).

```
root@primary1:~# cd /root/epic/temp
root@primary1:~/epic/temp# chmod +x preupgrade.sh
```

5. Execute the **preupgrade.sh** script to back up OpenSearch and upgrade Paragon Shell to release 2.4.X.

```
root@primary1:~/epic/temp# ./preupgrade.sh
```

Paragon Shell is upgraded to release 2.4.X on the primary node only.

6. Type `cli` to log in to Paragon Shell.
7. (Optional) Manually back up OpenSearch data. The upgrade command used to upgrade the cluster backs-up critical OpenSearch data. If you want to back up all OpenSearch data, use the following command before upgrading.

```
root@primary1> request paragon opensearch-backup
```

OpenSearch data is backed up. You can now upgrade the cluster. Go to ["Upgrade the Paragon Automation Cluster" on page 75](#).

## Upgrade the Paragon Automation Cluster

### IN THIS SECTION

- Upgrade using the `local` Option | 75
- Upgrade using the `remote url` Option | 78

You can upgrade your Paragon Automation Release 2.2.0 or release 2.3.0 installation and all the applications running on it using *any one* of the following two options:

### Upgrade using the `local` Option

Use this option when your Paragon Automation installation is in an air-gapped environment with no access to the Internet. However, you need to be able to copy the **upgrade\_paragon-release-build-id.tgz** and **upgrade\_paragon-release-build-id.tgz.psig** files to your primary node. To upgrade using the `local` filename option:

1. Type `exit` to exit to the Linux root shell from Paragon Shell of the installer primary node.
2. Copy the **upgrade\_paragon-release-build-id.tgz** and **upgrade\_paragon-release-build-id.tgz.psig** files, of the version to which you want to upgrade, to the `/root/epic/temp` folder.

You might need to download the **upgrade\_paragon-release-build-id.tgz** and **upgrade\_paragon-release-build-id.tgz.psig** files from the *Juniper Software Download* site to your local computer before copying it to the primary node.

3. (Optional) Use the `gpg --verify` command to validate the digital signature of the upgrade file. For example:

```
root@primary1:~/epic/temp# gpg --verify upgrade_paragon-
release-2.4.X.8952.gbef82aec6b.tgz.psig upgrade_paragon-release-2.4.X.8952.gbef82aec6b.tgz
gpg: Signature made Tue Feb 23 01:00:09 2024 UTC
gpg: using RSA key 4B7B22C9C4FE32CF
gpg: Good signature from "Northstar Paragon Automation 2024 ca@juniper.net" [ultimate]
```

Here `primary1` is the installer primary node. Validation takes a couple of minutes to complete.

4. Type `cli` to re-enter Paragon Shell.
5. Use the following command to upgrade the Paragon Automation cluster:

```
request paragon cluster upgrade local filename upgrade_paragon-release-build-id.tgz
```

For example:

```
root@primary1> request paragon cluster upgrade local filename upgrade_paragon-
release-2.4.X.8952.gbef82aec6b.tgz
Checking paragon cluster system health before proceeding with cluster upgrade. This will take
a minute...
...
<output snipped>
...
=====
Overall cluster status
=====

GREEN

=====

Paragon cluster is healthy.
Proceed with Paragon cluster upgrade.
Upgrade is in progress ...
Updated to build: paragon-release-2.4.X.8952.gbef82aec6b
Paragon Cluster upgrade is successful!
Run 'request paragon health-check' command to check current system health with upgraded
Paragon cluster.
Please continue to primary host node to upgrade Paragon-shell and update OVA system files by:
/root/epic/upgrade_paragon-shell_ova-system.sh
```

Here primary1 is the installer primary node. The upgrade command checks the health of the cluster before upgrading. If the cluster health check returns a GREEN status, the cluster is upgraded requiring no further input. If the cluster health check returns a RED status, the cluster is not upgraded. If the cluster health check returns an AMBER status, you are prompted to choose to continue or stop the upgrade.

**a.** no-confirm—Usage example:

```
request paragon cluster upgrade local filename upgrade_paragon-release-build-id.tgz no-confirm
```

Use the no-confirm option to ignore the AMBER status and continue with the upgrade without being prompted. However, the no-confirm option does not ignore a RED status.

**b. detach-process—Usage example:**

```
request paragon cluster upgrade local filename upgrade_paragon-release-build-id.tgz detach-process
```

As the upgrade process takes over an hour to complete, you can let the upgrade run in the background and free up the CLI screen for any other tasks. The command runs the initial health checks and then proceeds with the upgrade. Once the upgrade process starts, the process is detached and moved into the background and you are returned to the command prompt. The upgrade output is logged in the **/epic/temp/upgrade.log** file. To monitor the status of the upgrade process and print the output onscreen, use the `monitor start /epic/temp/upgrade.log` command. When the upgrade process completes, a success message similar to the following is displayed on all the cluster nodes:

Paragon Cluster upgrade is successful!

- Run 'request paragon health-check' command to check current system health with upgraded Paragon cluster.

- Please continue to primary host node to upgrade Paragon-shell and update OVA system files by:  
/root/epic/upgrade\_paragon-shell\_ova-system.sh

If you get disconnected from the VM during the upgrade process, you can periodically check the upgrade log file for status on the upgrade.

**c. input—Usage example:**

```
request paragon cluster upgrade local filename upgrade_paragon-release-build-id.tgz input input-string
```

Use the `input` option to pass additional Ansible input parameters to the upgrade command. For example, if you want to enable verbose logging during upgrade, use the `-v` option.

```
request paragon cluster upgrade local filename upgrade_paragon-release-build-id.tgz input "-v"
```

Your Paragon Automation installation and all the applications running on it are upgraded.

Note that, the upgrade process takes over an hour to complete.

6. Execute the `request paragon health-check` command to ensure that the upgraded cluster is healthy and operational.

The Overall Cluster Status must be GREEN.

For example:

```

root@primary1> request paragon health-check
Health status checking...

=====
Get node count of Kubernetes cluster.
=====

OK
There are 4 nodes in the cluster.
...
<output snipped>
...

=====
Overall cluster status
=====

GREEN

```

7. Upgrade Paragon Shell and the OVA system files. While Paragon Shell is upgraded on the installer primary node already, you must upgrade it on all the cluster nodes.

Go to ["Upgrade Paragon Shell and the OVA System Files" on page 82](#).

## Upgrade using the remote url Option

Use this option if your Paragon Automation installation has access to the Internet and the upgrade file is in a remote location. To upgrade using the remote url option:

1. Use the following Paragon Shell command on the installer primary node to upgrade the Paragon Automation cluster:

```
request paragon cluster upgrade remote url "https://juniper.software.download.site/upgrade_paragon-release-build-id.tgz?query_string"
```

For example:

```

root@primary1> request paragon cluster upgrade remote url "https://cdn.juniper.net/software/paragon-images/upgrade_paragon-release-2.3.0.8213.g458486e9da.tgz?query_string"
Checking paragon cluster system health before proceeding with cluster upgrade. This will take a minute...

```

```

...
<output snipped>
...
=====
Overall cluster status
=====

GREEN

=====

Paragon cluster is healthy.
Proceed with Paragon cluster upgrade.
Upgrading paragon cluster from https://cdn.juniper.net/software/paragon-images
Downloading tarball file upgrade_paragon-release-2.4.X.8952.gbef82aec6b
Download file size: 28,831,677,064 bytes
Current disk Usage:
 Total: 263,622,004,736 bytes
 Used: 106,109,399,040 bytes
 Available: 145,685,159,936 bytes
Please wait for current download to finish... (File is large. It may take a while.)
Upgrade tarball file is downloaded.
Upgrade is in progress ...
Updated to build: eop-release-2.4.X.8952.gbef82aec6b
Paragon Cluster upgrade is successful!
Run 'request paragon health-check' command to check current system health with upgraded
Paragon cluster.
Please continue to primary host node to upgrade Paragon-shell and update OVA system files by:
/root/epic/upgrade_paragon-shell_ova-system.sh

```

Here primary1 is the installer primary node from which your cluster was originally installed. The upgrade command checks the health of the cluster before upgrading. If the cluster health check returns a GREEN status, the cluster is upgraded requiring no further input. If the cluster health check returns a RED status, the cluster is not upgraded. If the cluster health check returns an AMBER status, you are prompted to choose to continue or stop the upgrade.

#### **Additional upgrade command options:**

You can also use any one or more of the following command options along with the upgrade command while upgrading:

a. no-confirm—Usage example:

```
request paragon cluster upgrade remote url "https://juniper.software.download.site/upgrade_paragon-release-build-id.tgz?query_string" no-confirm
```

Use the `no-confirm` option to ignore the AMBER status and continue with the upgrade without being prompted. However, the `no-confirm` option does not ignore a RED status.

b. detach-process—Usage example:

```
request paragon cluster upgrade remote url "https://juniper.software.download.site/upgrade_paragon-release-build-id.tgz?query_string" detach-process
```

As the upgrade process takes over an hour to complete, you can let the upgrade run in the background and free up the CLI screen for any other tasks. The command runs the initial health checks and then proceeds with the upgrade. Once the upgrade process starts, the process is detached and moved into the background and you are returned to the command prompt. The upgrade output is logged in the `/epic/temp/upgrade.log` file. To monitor the status of the upgrade process and print the output onscreen, use the `monitor start /epic/temp/upgrade.log` command. When the upgrade process completes, a success message similar to the following is displayed on all the cluster nodes:

```
Paragon Cluster upgrade is successful!

- Run 'request paragon health-check' command to check current system health with upgraded
Paragon cluster.

- Please continue to primary host node to upgrade Paragon-shell and update OVA system
files by:
/root/epic/upgrade_paragon-shell_ova-system.sh
```

c. disk-saving—Usage example:

```
request paragon cluster upgrade remote url "https://juniper.software.download.site/upgrade_paragon-release-build-id.tgz?query_string" disk-saving
```

Use this option to delete the `upgrade_paragon-release-build-id.tgz` file as soon as it is unzipped from the primary node. The upgrade command downloads the upgrade file from the remote location and extracts the contents of the file at the beginning of the upgrade process. This option deletes the downloaded file as soon as it is unzipped to free up space on the primary node.

The advantage of using this option is that you need lesser free space for the upgrade process. The default minimum free space required is 15% of the total disk space + three times the upgrade file size. With this option you need a minimum free space of 15% of the total disk space + two times the upgrade file size.

**d. input—Usage example:**

```
request paragon cluster upgrade remote url "https://juniper.software.download.site/upgrade_paragon-
release-build-id.tgz?query_string" input input-string
```

Use the input option to pass additional Ansible input parameters to the upgrade command. For example, if you want to enable verbose logging during upgrade use the -v option.

```
request paragon cluster upgrade remote url "https://juniper.software.download.site/upgrade_paragon-release-
build-id.tgz?query_string" input "-v"
```

Your Paragon Automation installation and all the applications running on it are upgraded.

Note that, the upgrade process takes over an hour to complete.

2. Execute the request paragon health-check command to ensure that the upgraded cluster is healthy and operational.

The Overall Cluster Status must be GREEN.

For example:

```
root@primary1> request paragon health-check
Health status checking...

=====
Get node count of Kubernetes cluster.
=====

OK
There are 4 nodes in the cluster.
...
<output snipped>
...

=====
Overall cluster status
=====
```

GREEN

3. Upgrade Paragon Shell and the OVA system files. While Paragon Shell is upgraded on the installer primary node already, you must upgrade it on all the cluster nodes.

Go to ["Upgrade Paragon Shell and the OVA System Files" on page 82.](#)

## Upgrade Paragon Shell and the OVA System Files

When your Paragon Automation installation and all the applications running on it are successfully upgraded, you must upgrade Paragon Shell and the OVA system files.

1. Exit from the installer primary node Paragon Shell to the Linux root shell by typing `exit`.
2. Execute the Paragon Shell upgrade shell script.

```
root@primary1:~# bash /root/epic/upgrade_paragon-shell_ova-system.sh
Upgrading paragon-shell...
Updating paragon-shell for primary1.....
Container paragon-shell Stopping
Container paragon-shell Stopped
Container paragon-shell Removing
Container paragon-shell Removed
paragon-shell Pulling

...
<output snipped>
...

primaryname update-status
primary1 ok
primary3 ok
primary2 ok
primary4 ok
paragon-shell upgrade successful!
Updating OVA system files...
OVA system files update successful!
```

Paragon Shell and the OVA system files are upgraded.

3. (Optional) Check the build and OVA version of your upgraded setup from Paragon Shell.

```
root@primary> show paragon version
ova: 20250226_1117_ova
ova-patch: 20250226_0349
build: eop-2.4.X.8952.gbef82aec6b
Client Version: v1.29.6
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Server Version: v1.31.4+rke2r1
```

Now proceed to perform the post cluster upgrade tasks.

## Post Cluster Upgrade Tasks

After upgrading the cluster and Paragon Shell OVA, perform the following tasks to complete the upgrade process.

1. Update the base OS. See ["Update the OS" on page 85](#).
2. Upgrade the service designs, update the network implementation plan, and recreate the resource and service instances. See *Update the Network Implementation Plan and Recreate Service Instances After Upgrade*.

### RELATED DOCUMENTATION

[Update the OS | 85](#)

*Update the Network Implementation Plan and Recreate Service Instances After Upgrade*

[Back Up and Restore Paragon Automation | 103](#)

# 5

CHAPTER

## Manage the Cluster

---

### IN THIS CHAPTER

- [Update the OS | 85](#)
  - [Repair or Replace Cluster Nodes | 87](#)
  - [Shut Down and Reboot Nodes | 95](#)
  - [Increase TimescaleDB PVC Size | 97](#)
-

# Update the OS

You create the node virtual machines (VMs) in a Paragon Automation cluster using the OVA (or OVF and .vmdk) software download files. The files are pre-packaged with all the utilities, OS, and software required to create the VMs. The VMs are created with Ubuntu 22.04.5 LTS (Jammy Jellyfish) Linux base OS. You might need to update the base OS to maintain the security, stability, performance, and compatibility of the Kubernetes cluster. Juniper provides you with the required OS update file to enable you to update the OS on your node VMs. The OS update functionality in Paragon Automation includes the following updates:

- Linux kernel update
- OpenSSL or OS security update
- Any third-party packages required by Paragon Automation
- All packages that are part of the base OS

To update the OS, perform the following steps:

1. Download the **paragon-ubuntu-22-04-update-*date*.tar.gz** and **paragon-ubuntu-22-04-update-*date*.tar.gz.psig** files from the Juniper Software Download site on to your computer or local installer VM.
2. Log in to any one of the Paragon Automation cluster nodes as the root user.
3. Type `exit` to exit Paragon Shell to the Linux root shell.
4. Copy the **paragon-ubuntu-22-04-update-*date*.tar.gz** and **paragon-ubuntu-22-04-update-*date*.tar.gz.psig** files to the cluster node.
5. Update the OS on the current node using the copied files.

```
root@node# update-os paragon-ubuntu-22-04-update-date.tar.gz paragon-ubuntu-22-04-update-date.tar.gz.psig
```

The process takes around 15-30 minutes to complete depending on the size of the updated packages. Once complete, the OS on the other three nodes of the cluster is also automatically updated.

## Verify the OS update

You can perform any of the following steps to verify that the OS update process is successful, and the cluster operation is unaffected.

- The Paragon Automation cluster remains operational while updating the OS. To verify if the update process has succeeded, check the `/var/log/apt/history.log` log file to see the timestamp of the last update and updated packages. For example:

```
Start-Date: 2024-09-18 15:50:57
```

```
Commandline: apt upgrade
```

```
Install: ubuntu-pro-client-l10n:amd64 (31.2.2~22.04, automatic), ubuntu-pro-client:amd64 (31.2.2~22.04, automatic)
```

```
Upgrade: dpkg:amd64 (1.21.1ubuntu2.1, 1.21.1ubuntu2.3), libxtables12:amd64 (1.8.7-1ubuntu5, 1.8.7-1ubuntu5.2), initramfs-tools-core:amd64 (0.140ubuntu13.1, 0.140ubuntu13.4), udev:amd64 (249.11-0ubuntu3.7, 249.11-0ubuntu3.12), coreutils:amd64 (8.32-4.1ubuntu1, 8.32-4.1ubuntu1.2), libmm-glib0:amd64 (1.20.0-1~ubuntu22.04.1, 1.20.0-1~ubuntu22.04.3), python3-tz:amd64 (2022.1-1ubuntu0.22.04.0, 2022.1-1ubuntu0.22.04.1), openssh-client:amd64 (1:8.9p1-3ubuntu0.6, 1:8.9p1-3ubuntu0.7), iptables:amd64 (1.8.7-1ubuntu5, 1.8.7-1ubuntu5.2), python3-distupgrade:amd64 (1:22.04.16, 1:22.04.19), apt:amd64 (2.4.8, 2.4.12), sosreport:amd64 (4.4-1ubuntu1.22.04.1, 4.5.6-0ubuntu1~22.04.2), cryptsetup-bin:amd64 (2:2.4.3-1ubuntu1.1, 2:2.4.3-1ubuntu1.2), git:amd64 (1:2.34.1-1ubuntu1.9, 1:2.34.1-1ubuntu1.10), libunwind8:amd64 (1.3.2-2build2, 1.3.2-2build2.1), libldap-common:amd64 (2.5.16+dfsg-0ubuntu0.22.04.2, 2.5.17+dfsg-0ubuntu0.22.04.1), ufw:amd64 (0.36.1-4build1, 0.36.1-4ubuntu0.1), sg3-utils:amd64 (1.46-1build1, 1.46-1ubuntu0.22.04.1), grub-pc-bin:amd64 (2.06-2ubuntu7.1, 2.06-2ubuntu7.2), libfwupd2:amd64 (1.7.9-1~22.04.1, 1.7.9-1~22.04.3), libapt-pkg6.0:amd64 (2.4.8, 2.4.12), initramfs-tools-bin:amd64 (0.140ubuntu13.1, 0.140ubuntu13.4), apparmor:amd64 (3.0.4-2ubuntu2.2, 3.0.4-2ubuntu2.3), libip4tc2:amd64 (1.8.7-1ubuntu5, 1.8.7-1ubuntu5.2), libapparmor1:amd64 (3.0.4-2ubuntu2.2, 3.0.4-2ubuntu2.3), openssh-server:amd64 (1:8.9p1-3ubuntu0.6, 1:8.9p1-3ubuntu0.7), irqbalance:amd64 (1.8.0-1build1, 1.8.0-1ubuntu0.2), python-apt-common:amd64 (2.4.0ubuntu1, 2.4.0ubuntu3), libgpgme11:amd64 (1.16.0-1.2ubuntu4, 1.16.0-1.2ubuntu4.2), libldap-2.5-0:amd64 (2.5.16+dfsg-0ubuntu0.22.04.2, 2.5.17+dfsg-0ubuntu0.22.04.1), libudev1:amd64 (249.11-0ubuntu3.7, 249.11-0ubuntu3.12), motd-news-config:amd64 (12ubuntu4.3, 12ubuntu4.6), libsgutils2-2:amd64 (1.46-1build1, 1.46-1ubuntu0.22.04.1), libc6:amd64 (2.35-0ubuntu3.6, 2.35-0ubuntu3.7), locales:amd64 (2.35-0ubuntu3.6, 2.35-0ubuntu3.7), fwupd-signed:amd64 (1.51~22.04.1+1.2-3ubuntu0.2, 1.51.1~22.04.1+1.4-0ubuntu0.1), cloud-init:amd64 (23.1.2-0ubuntu0~22.04.1, 23.4.4-0ubuntu0~22.04.1), sg3-utils-udev:amd64 (1.46-1build1, 1.46-1ubuntu0.22.04.1), open-vm-tools:amd64 (2:12.1.5-3~ubuntu0.22.04.4, 2:12.3.5-3~ubuntu0.22.04.1), base-files:amd64 (12ubuntu4.3, 12ubuntu4.6), mdadm:amd64 (4.2-0ubuntu1, 4.2-0ubuntu2), cryptsetup-initramfs:amd64 (2:2.4.3-1ubuntu1.1, 2:2.4.3-1ubuntu1.2), python3-apt:amd64 (2.4.0ubuntu1, 2.4.0ubuntu3), snapd:amd64 (2.58+22.04.1, 2.61.3+22.04), systemd-hwe-hwdb:amd64 (249.11.3, 249.11.5), python3-distro-info:amd64 (1.1build1, 1.1ubuntu0.2), libcryptsetup12:amd64 (2:2.4.3-1ubuntu1.1,
```

```
2:2.4.3-1ubuntu1.2), multipath-tools:amd64 (0.8.8-1ubuntu1.22.04.1, 0.8.8-1ubuntu1.22.04.4),
distro-info-data:amd64 (0.52ubuntu0.2, 0.52ubuntu0.6), libip6tc2:amd64 (1.8.7-1ubuntu5,
1.8.7-1ubuntu5.2), grub2-common:amd64 (2.06-2ubuntu7.1, 2.06-2ubuntu7.2), cryptsetup:amd64
(2:2.4.3-1ubuntu1.1, 2:2.4.3-1ubuntu1.2), distro-info:amd64 (1.1build1, 1.1ubuntu0.2),
openssh-sftp-server:amd64 (1:8.9p1-3ubuntu0.6, 1:8.9p1-3ubuntu0.7), grub-common:amd64
(2.06-2ubuntu7.1, 2.06-2ubuntu7.2), ubuntu-standard:amd64 (1.481, 1.481.1), libc-bin:amd64
(2.35-0ubuntu3.6, 2.35-0ubuntu3.7), http://netplan.io :amd64 (0.105-0ubuntu2~22.04.3,
0.106.1-7ubuntu0.22.04.2), python3-apport:amd64 (2.20.11-0ubuntu82.4, 2.20.11-0ubuntu82.5),
initramfs-tools:amd64 (0.140ubuntu13.1, 0.140ubuntu13.4), ubuntu-server:amd64 (1.481,
1.481.1), apt-utils:amd64 (2.4.8, 2.4.12), ubuntu-release-upgrader-core:amd64 (1:22.04.16,
1:22.04.19), libfwupdplugin5:amd64 (1.7.9-1~22.04.1, 1.7.9-1~22.04.3), ubuntu-advantage-
tools:amd64 (27.13.6~22.04.1, 31.2.2~22.04), ethtool:amd64 (1:5.16-1, 1:5.16-1ubuntu0.1), git-
man:amd64 (1:2.34.1-1ubuntu1.9, 1:2.34.1-1ubuntu1.10), grub-pc:amd64 (2.06-2ubuntu7.1,
2.06-2ubuntu7.2), kpartx:amd64 (0.8.8-1ubuntu1.22.04.1, 0.8.8-1ubuntu1.22.04.4), python3-
problem-report:amd64 (2.20.11-0ubuntu82.4, 2.20.11-0ubuntu82.5), libnetplan0:amd64
(0.105-0ubuntu2~22.04.3, 0.106.1-7ubuntu0.22.04.2), apport:amd64 (2.20.11-0ubuntu82.4,
2.20.11-0ubuntu82.5), ubuntu-minimal:amd64 (1.481, 1.481.1), update-notifier-common:amd64
(3.192.54.5, 3.192.54.8), python3-debian:amd64 (0.1.43ubuntu1, 0.1.43ubuntu1.1)
```

End-Date: 2024-09-18 15:52:27

- Verify that cluster-operation is unaffected by checking that all pods are in Running status using the following command in Paragon Shell.

```
> show paragon cluster pods
```

- Verify that the cluster is healthy and operational using the following command in the Paragon Shell.

```
> request paragon health-check
```

## Repair or Replace Cluster Nodes

### IN THIS SECTION

● [Repair Nodes | 88](#)

- [Replace Faulty Nodes | 91](#)

You can repair and replace faulty nodes in your Paragon Automation cluster using Paragon Shell. This topic describes how to repair and replace nodes in your cluster.

## Repair Nodes

### IN THIS SECTION

- [Repair node failure scenario | 90](#)

To repair a faulty node in your existing Paragon Automation cluster.

1. Log in to the Linux root shell of the faulty node.

If you are unable to log in to the faulty node, go to step ["6" on page 89](#).

2. Stop and kill all RKE2 services on the faulty node.

```
root@node-f:~# rke2-killall.sh
```

If you see the following error, reboot the node using the `shutdown -R now` command.

```
rm: cannot remove '/var/lib/kubelet/plugins/kubernetes.io/csi/rook-ceph.cephfs.csi.ceph.com/aadbbffbb9d894e02c7c7a2510fe5fd72e28d2fe4a23565d764d84176f055d62/globalmount': Device or resource busy
```

3. Uninstall RKE2.

```
root@node-f:~# rke2-uninstall.sh
```

4. Reboot the node using the `shutdown -r now` command.

5. Clear the data on the disk partition used for Ceph.

```
root@node-f:~# wipefs -a -f /dev/partition
root@node-f:~# dd if=/dev/zero of=/dev/partition bs=1M count=100
```

Use **/dev/sdb** if you used the OVA bundle to deploy your cluster on an ESXi server. Use **/dev/vdb** if you deployed on Proxmox VE.

6. Log in to the Linux root shell of the node you deployed the cluster from. Note that you can repair the faulty node from any functional node in the cluster.

If the installer node is faulty, then perform the following steps.

- a. Log in to one of the other two primary nodes (that is nodes with Kubernetes index 2 or 3).
  - b. Type `exit` to exit to the Linux root shell.
  - c. Edit the **inventory** file to reorder the node IP addresses list. Use a text editor, to move the node that you logged in to, to the top of the list of the three IP addresses displayed under the `master` section.
7. Delete the faulty node from the cluster.

```
root@primary1:~# kubectl delete nodes faulty-node-hostname
```

Where *faulty-node-hostname* is the hostname of the node you want to repair.

8. Type `cli` to enter Paragon Shell.

If you are not logged in to the node from which you deployed the cluster, then log out of the current node, and log in to the installer node.

9. Repair the node from Paragon Shell.

```
root@primary1> request paragon repair-node address ip-address-of-faulty-node
```

Where *ip-address* is the IP address of the node that you want to repair.

10. (Optional) Ensure that the cluster is healthy. Type `request paragon health-check` to determine cluster health.

## Repair node failure scenario

You might encounter issues with node repair, where the repair procedure is executed successfully, but rook-ceph-osd is in `Init:CrashLoopBackOff` state on the repaired node. For example:

```
root@primary3> show paragon cluster pods |grep rook-ceph-osd
rook-ceph rook-ceph-osd-0-c744fd5f9-jgnlr 2/2 Running
0 5h39m
rook-ceph rook-ceph-osd-1-c8f7c8c4f-k98cr 2/2 Running
0 5h39m
rook-ceph rook-ceph-osd-2-9957b89f-jv6nf 0/2 Init:CrashLoopBackOff
22 (81s ago) 142m
rook-ceph rook-ceph-osd-3-77f7b7f548-26d67 2/2 Running
0 5h38m
rook-ceph rook-ceph-osd-prepare-Primary1 0/1 Completed
0 100m
rook-ceph rook-ceph-osd-prepare-Primary2 0/1 Completed
0 100m
rook-ceph rook-ceph-osd-prepare-Worker4 0/1 Completed
0 99m
```

Here the failed OSD number is 2.

Perform the following additional steps to fix the issue.

1. Stop the rook-ceph-operator and set replica to 0.

```
root@primary3:~# kubectl -n rook-ceph scale deployment rook-ceph-operator --replicas=0
```

2. Remove the failing rook-ceph-osd-*osd-number* OSD processes. For example:

```
root@primary3:~# kubectl delete deploy -n rook-ceph rook-ceph-osd-2
```

3. Connect to the toolbox.

```
root@primary1:~# kubectl exec -ti -n rook-ceph $(kubectl get po -n rook-ceph -l app=rook-ceph-tools -o jsonpath={..metadata.name}) -- bash
```

4. Mark out the failed *osd-number* OSD. For example:

```
bash-4.4$ ceph osd out 2
```

5. Remove the failed *osd-number* OSD. For example:

```
bash-4.4$ ceph osd purge 2 --yes-i-really-mean-it
bash-4.4$ exit
```

6. Clear the data on the disk partition used for Ceph.

```
root@primary3:~# wipefs -a -f /dev/vdb
root@primary3:~# dd if=/dev/zero of=/dev/vdb bs=1M count=100
```

Use **/dev/sdb** if you used the OVA bundle to deploy your cluster on an ESXi server. Use **/dev/vdb** if you deployed on Proxmox VE.

7. Reboot the node.

```
root@primary3:~# shutdown -r now
```

8. Restart the rook-ceph-operator.

```
root@primary3:~# kubectl -n rook-ceph scale deployment rook-ceph-operator --replicas=1
```

## Replace Faulty Nodes

### IN THIS SECTION

- [Replace-node failure scenario | 94](#)

You can replace a faulty node with a replacement node. To replace a node, you must prepare the replacement node, delete the faulty node and then add the replacement node to the cluster.

1. Log in to the Linux root shell of one of the functional nodes of the cluster and delete the faulty node.

```
root@primary1:~# kubectl delete node faulty-node-hostname
```

Where *faulty-node-hostname* is the hostname of the node you want to replace.

## 2. Prepare the replacement node.

You must create and configure the replacement node before replacing a faulty node.

The node which is replacing the faulty node can have a new IP address or the same IP address as the faulty node, but you will still need to create and configure the node VM.

- a. To create the node, depending on your hypervisor server, perform the steps detailed in ["Create the Node VMs" on page 27](#).
- b. Once the node is created, configure the node as detailed in ["Configure the node VMs" on page 40](#).

## 3. Replace the Faulty Node:

Once the replacement node is prepared, log in to the node from which you deployed your existing Paragon Automation cluster. You are placed in Paragon Shell.

4. If the IP address of the replacement node is same as the IP address of the faulty node, go to step ["5" on page 94](#). If the IP address of the replacement node is different from the IP address of the faulty node, perform the following steps.
  - a. To edit the cluster, type `configure` to enter the configuration mode.

```
root@primary1> configure
Entering configuration mode

[edit]
root@primary1#
```

- b. Remove the faulty node using the `delete paragon cluster nodes kubernetes x` command.

Where *x* is the index number of the node that you want to remove. For example, if the node with Kubernetes index 3 is faulty, use:

```
root@primary1# delete paragon cluster nodes kubernetes 3
```

- c. Add the replacement node to the cluster configuration in place of the node you removed using the `set paragon cluster nodes kubernetes x address node-x-new-IP` command.

Where  $x$  is the index number of the faulty node and *node- $x$ -new-IP* is the IP address of the replacement node. For example, if 10.1.2.11 is the IP address of the replacement node and if the node with Kubernetes index 3 is faulty:

```
root@primary1# set paragon cluster nodes kubernetes 3 address 10.1.2.11

root@primary1# commit
commit complete
```

d. Commit the configuration.

```
root@primary1# commit
commit complete
```

e. (Optional) Verify the cluster configuration.

```
root@primary1# show paragon cluster nodes
kubernetes 1 {
 address 10.1.2.3;
}
kubernetes 2 {
 address 10.1.2.4;
}
kubernetes 3 {
 address 10.1.2.11;
}
kubernetes 4 {
 address 10.1.2.6;
}
```

f. Exit configuration mode and regenerate the configuration files.

```
root@primary1# exit
Exiting configuration mode

root@primary1> request paragon config
Paragon inventory file saved at /epic/config/inventory
Paragon config file saved at /epic/config/config
```

## 5. Regenerate SSH keys on the cluster nodes.

When prompted, enter the SSH password for all the existing VMs and the new VM. Enter the same password that you configured to log in to the VMs.

```
root@primary1> request paragon ssh-key
Please enter comma-separated list of IP addresses: 10.1.2.3,10.1.2.4,10.1.2.6,10.1.2.11
Please enter SSH username for the node(s): root
Please enter SSH password for the node(s): password
checking server reachability and ssh connectivity ...
Connectivity ok for 10.1.2.3
Connectivity ok for 10.1.2.4
Connectivity ok for 10.1.2.6
Connectivity ok for 10.1.2.11

<output snipped>
```

## 6. Replace the node.

```
root@primary1> request paragon replace-node address 10.1.2.11
Process running with PID: 23xx032
To track progress, run 'monitor start /epic/config/log'
```

Where *10.1.2.11* is the IP address of the replacement node.

## 7. (Optional) Ensure that the cluster is healthy. Type `request paragon health-check` to determine cluster health.

### Replace-node failure scenario

You might encounter issues with node-replacement when the IP address (or hostname) of the replacement node is different from the IP address (or hostname) of the faulty node; perform the following additional steps to fix the issue.

1. Log in to the Linux root shell of the node from where you deployed the cluster.
2. Delete the local volume pvc associated with the faulty node, if any.

```
root@primary1:~# paragon-remove-object-from-deleted-node -t faulty-node-hostname -y
```

3. Run the following command to check if the status of Rook and Ceph pods installed in the rook-ceph namespace is not Running or Completed.

```
root@primary1:~# kubectl get po -n rook-ceph
```

4. Remove the failing OSD processes.

```
root@primary1:~# kubectl delete deploy -n rook-ceph rook-ceph-osd-number
```

5. Connect to the toolbox.

```
root@primary1:~# kubectl exec -ti -n rook-ceph $(kubectl get po -n rook-ceph -l app=rook-ceph-tools -o jsonpath={..metadata.name}) -- bash
```

6. Identify the failing OSD.

```
bash-4.4$ ceph osd status
```

7. Mark out the failed OSD.

```
bash-4.4$ ceph osd out osd-ID-number
```

8. Remove the failed OSD.

```
bash-4.4$ ceph osd purge osd-ID-number --yes-i-really-mean-it
```

## SEE ALSO

[Install Paragon Automation](#) | 22

# Shut Down and Reboot Nodes

You may need to shut down or reboot one or more or all the node VMs for reasons such as performing a scheduled maintenance activity, upgrading your ESXi server, recovering from a node failure, and so on. This topic describes how to gracefully shut down and restart your node VMs and the entire Paragon Automation cluster.

### Shut down a node VM

1. Log in to the node VM. You are placed in Paragon Shell.
2. (Optional) Check the health of your cluster. Execute the request `paragon health-check` command. The Overall Cluster Status must be GREEN or AMBER.
3. Shut down the node using the request `paragon shutdown type node` command.

### Reboot a node VM

1. Log in to the node VM. You are placed in Paragon Shell.
2. Reboot the node using the request `paragon reboot type node` command.
3. (Optional) After the VM has rebooted, verify that the cluster is in good health. Execute the request `paragon health-check` command. The Overall Cluster Status must be GREEN or AMBER.

### Shut down the cluster

1. Log in to any node VM. You are placed in Paragon Shell.
2. (Optional) Check the health of your cluster. Execute the request `paragon health-check` command. The Overall Cluster Status must be GREEN or AMBER.
3. Shut down the cluster using the request `paragon shutdown type cluster` command.

### Reboot the cluster

1. Log in to the node VM. You are placed in Paragon Shell.
2. Reboot the node using the request `paragon reboot type cluster` command.
3. (Optional) After the cluster has rebooted, verify that the cluster is in good health. Execute the request `paragon health-check` command. The Overall Cluster Status must be GREEN or AMBER.

# Increase TimescaleDB PVC Size

## SUMMARY

Use the steps detailed in this topic to manually increase the size for TimescaleDB persistent volume claim (PVC).

## IN THIS SECTION

- [Increase Ceph Storage | 97](#)
- [Update Timescale DB PVC Quota | 100](#)
- [Increase Timescale DB PVC | 100](#)

The minimum recommended system requirements to deploy a Paragon Automation cluster is described in ["Paragon Automation System Requirements" on page 10](#). If you want to scale up your cluster you can increase the hardware resources that are required on each node VM when you are creating the VMs for the first time as per your requirement. If you have already deployed your cluster, use the information detailed in this topic to increase storage size, specifically Ceph and consequently TimescaleDB PVC.

Ceph storage is used for both object storage and the PVC for the pods. ~50% of Ceph storage is allocated for TimescaleDB PVC.

To increase the size of the PVC, perform the following steps:

1. ["Increase Ceph storage." on page 97](#)
2. ["Update the allocation quota for Timescale DB PVC in Ceph storage." on page 100](#)
3. ["Increase the Timescale DB PVC size." on page 100](#)

## Increase Ceph Storage

Increase the total Ceph storage size.

1. Increase the virtual disk size from your ESXi server.

Ceph uses the second disk attached to the node virtual machine (VMs). Ensure that you resize the correct virtual disk. You do not need to reboot the VM.
2. Verify the new virtual disk size.
  - a. Log in to the Linux root shell of the node VM.

- b. Use the `lsblk` command to verify that the OS detects the new disk size. For example:

```
root@vm4:~# lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
loop0 7:0 0 64M 1 loop /snap/core20/2379
loop1 7:1 0 63.7M 1 loop /snap/core20/2434
loop2 7:2 0 87M 1 loop /snap/lxd/29351
loop3 7:3 0 89.4M 1 loop /snap/lxd/31333
loop4 7:4 0 38.8M 1 loop /snap/snapd/21759
loop5 7:5 0 44.3M 1 loop /snap/snapd/23258
sr0 11:0 1 4M 0 rom
nbd0 43:0 0 0B 0 disk
nbd1 43:32 0 0B 0 disk
nbd2 43:64 0 0B 0 disk
nbd3 43:96 0 0B 0 disk
nbd4 43:128 0 0B 0 disk
nbd5 43:160 0 0B 0 disk
nbd6 43:192 0 0B 0 disk
nbd7 43:224 0 0B 0 disk
vda 252:0 0 700G 0 disk
└─vda1 252:1 0 1M 0 part
└─vda2 252:2 0 700G 0 part /var/lib/kubelet/pods/fde8c46d-f069-4203-bd4e-3897d5915559/
volume-subpaths/config/network/0
....
 /export/local-volumes/pv1
 /
vdb 252:16 0 55G 0 disk
```

In this example, the OS detects that the `vdb` was increased from 50-GB to 55-GB.

3. Restart Ceph OSD to detect the size change. For example:

- a. Verify the existing OSD size.
- i. Launch the Rook tools pod.

```
root@vm1:~# kubectl exec -ti -n rook-ceph $(kubectl get po -n rook-ceph -l app=rook-
ceph-tools -o jsonpath={..metadata.name}) -- bash
```

- ii. Retrieve the current OSD size.

```
bash-4.4$ ceph osd status
```

ID	HOST	USED	AVAIL	WR OPS	WR DATA	RD OPS	RD DATA	STATE
0	vm2	2846M	47.2G	0	2633	0	0	exists,up
1	vm3	3132M	46.9G	1	28.6k	0	1135k	exists,up
2	vm4	<b>3065M</b>	<b>47.0G</b>	4	23.9k	3	201k	exists,up
3	vm1	2897M	47.1G	1	2698	1	979k	exists,up

In this example, you are modifying the OSD on vm4. Here, we still see original size, which is ~50-GB (USED+AVAIL).

- b. Go back to the Linux root shell and determine the OSD pod that runs on the node for which the disk size is to be increased (vm4).

```
root@vm1:~# kubectl get pod -A -o wide | grep osd | grep vm4
```

		rook-ceph-osd-2-787df64c87-bkjt8	2/2
Running	2 (4d3h ago)	13d 10.1.2.8	vm4 <none> <none>

- c. Restart the pod.

```
root@vm1:~# kubectl rollout restart deploy -n rook-ceph rook-ceph-osd-2-787df64c87-bkjt8
```

- d. Verify the new size.

- i. Launch the Rook tools pod and execute `ceph osd status` again.

```
bash-4.4$ ceph osd status
```

ID	HOST	USED	AVAIL	WR OPS	WR DATA	RD OPS	RD DATA	STATE
0	vm2	2924M	47.1G	0	4403	0	0	exists,up
1	vm3	3202M	46.8G	0	426k	0	1140k	exists,up
2	vm4	<b>1474M</b>	<b>53.5G</b>	0	477	1	186k	exists,up
3	vm1	2977M	47.0G	3	21.4k	3	1008k	exists,up

Here, the OSD on vm4 has increased to ~55-GB.

- ii. Verify that the total size has also increased.

```
bash-4.4$ ceph status
...
data:
...
usage: 11 GiB used, 194 GiB / 205 GiB avail
```

Here, the total size has increased from 200-GB to 205-GB.



**NOTE:** In this example, you increased the size of Ceph storage on one node VM. We recommend that you increase the size on all the node VMs to make the storage value consistent across all the nodes. Repeat these steps for the remaining three VMs.

## Update Timescale DB PVC Quota

If you increase the total size of Ceph storage, you must update the allocation quota between object storage and PVC. Use Paragon Shell to increase the allocation quota. For example:

```
root@vm1> request paragon deploy cluster input "-t rook-quota"
Process running with PID: 1830232
To track progress, run 'monitor start /epic/config/log'
```

## Increase Timescale DB PVC

You can increase the PVC size for TimescaleDB by using Paragon Shell. When you are installing Paragon Automation afresh, override the default PVC size and add the set paragon cluster custom-config keyvalue paa-timescaledb-storage-size string *size* configuration before deploying the Paragon Automation cluster.

1. Log in to the installer node VM.
2. Configure the new PVC size in configure mode. For example:

```
root@vm1> configure
Entering configuration mode
```

```
[edit]
root@vm1# set paragon cluster custom-config keyValue paa-timescaledb-storage-size string 40Gi

[edit]
root@vm1# commit
warning: *** operating on 10.1.2.8 ***
warning: *** operation on 10.1.2.8 succeeds ***
warning: *** operating on 10.1.2.7 ***
warning: *** operation on 10.1.2.7 succeeds ***
warning: *** operating on 10.1.2.6 ***
warning: *** operation on 10.1.2.6 succeeds ***
commit complete

[edit]
root@vm1# exit
Exiting configuration mode

root@vm1> request paragon config
Paragon inventory file saved at /epic/config/inventory
Paragon config file saved at /epic/config/config.yml
```

3. Deploy the Paragon Automation cluster as usual.

# 6

CHAPTER

## Backup and Restore

---

### IN THIS CHAPTER

- [Back Up and Restore Paragon Automation | 103](#)
  - [Disaster Recovery using VMware ESXi | 110](#)
-

# Back Up and Restore Paragon Automation

## SUMMARY

This topic describes the backup and restore functionality available for Paragon Automation.

## IN THIS SECTION

- [Back Up Using Paragon Shell | 103](#)
- [Restore Using Paragon Shell | 106](#)
- [View or Delete Backup Files | 108](#)
- [Upload or Download Backup Files | 109](#)

You can use the backup and restore functionality available in Paragon Shell to back up and restore your Paragon Automation cluster and application configuration data.

## Back Up Using Paragon Shell

You can back up your current Paragon Automation network configuration using Paragon Shell CLI. When you run the backup command, all the application configuration information stored in PostgreSQL, ArangoDB, and Airflow configuration database systems, LLM connector secrets keys, and available software images are backed up. The backup command also backs up telemetry information stored in OpenSearch, TimescaleDB, and VictoriaMetrics database systems. The backup procedure can be performed while the microservices and applications are running and does not affect the operation of the network. However, we recommend that you do not perform a backup during configuration changes such as device onboarding.

To back up your Paragon Automation configuration state.

1. Log in as root user to any of the Paragon Automation nodes.
2. Execute the request `paragon backup` command to back up the configuration. For example:

```
root@Primary1> request paragon backup
Postgres backup started
Postgres backup completed
Using arango CRDN pod foghorn-crdn-e9jiiyh7-413778
ArangoDB backup completed
Airflow backup completed
Checking available software images
```

```

Downloading software images
Image download progress : 0%| | 0/1 [00:00]
Image download progress : 100%|#####| 1/1 [00:20]
Image download progress : 100%|#####| 1/1 [00:20]
Software image backup completed
AskParagon backup completed
Timescaledb backup job started
Opensearch backup job started
insights Victorimetrics backup job started
pathfinder Victorimetrics backup job started
Backup job in progress for 20250318-090214

```

The backup job runs in the background and you are returned to the command prompt. In this example, 20250318-090214 is the backup ID in the **yyyymmdd-hhmmss** format. The backup data is stored in a folder named as the backup ID.

Alternatively, you can use the `request paragon backup config` command to back up only configuration information or `request paragon backup telemetry` command to back up only telemetry data.

The backup command checks the space available in the local file system before taking a backup. If the node does not have enough storage space, the backup process will display an error and fail.

3. (Optional) View progress of the backup job. The backup process takes a few minutes to complete. In order to not block the terminal for use for the whole duration of the backup job, you are returned to the command prompt before the backup job is complete. You can view the progress status of the backup job by using the `show paragon backup status backup-id backupID` command.

For example:

```

root@Primary1> show paragon backup status backup-id 20250318-090214
Backup (config data) status for backup ID 20250318-090214:
 Paragon configuration : SUCCESS
Backup (telemetry data) status for backup ID 20250318-090214:
 Timescale job status : SUCCESS
 Opensearch job status : SUCCESS
 Insights Victorimetrics job status : SUCCESS
 Pathfinder Victorimetrics job status : SUCCESS

```

4. (Optional) If you want to store the backup in a remote location, upload the backup folder to the remote location using the following command:

```

root@Primary1> request paragon backup upload backup-id backupID storage-location remote-path
user username password password

```

Where:

*remote-path* is the path to the remote backup location.

*username* and *password* are the login credentials to the remote server.

Upon completion of the backup process, the backup folder is stored in the local persistent **/export/paragon-shell/backup** folder on the node. You'll have to exit out of Paragon Shell to the Linux root shell to navigate to the folder where the backups are saved.

Each backup folder contains the following folders with backup-related information.

- **airflow**—Backup of airflow secrets and DAGs
- **arango**—Backup of ArangoDB database
- **ask\_paragon**—Configured LLM connector secrets (in encrypted format)
- **postgres**—Backup of PostgresDB database
- **software\_images**—Backup of available devices images
- **system\_config**—Backup of system configuration; this is for reference only

Telemetry data backups are stored in the nodes where the database pods are running.

### Caveats of the backup process

- Application configurations (such as devices, sites, service orders, and so on) are backed up, but certificates and infrastructure services configurations are not backed up. This information must be kept unchanged before you perform a restore.
- The backup process captures the current infrastructure configurations, but information is used for reference only. The same configuration can be used to instantiate a new setup.

For example, if monitoring was enabled on the cluster before performing a backup. The configuration related to monitoring is stored in the **/export/paragon-shell/backup/backupID/system\_config/config.cmgd** file. Post-restore, use the information in the monitoring section of the **config.cmgd** file to reconfigure the monitoring commands on the new setup.

- Telemetry backups are not version controlled, that is, only the latest copy of a backup is available at any given point of time. A new backup command will overwrite the existing telemetry backup with the delta of the data from the last backup.

To maintain multiple and periodic copies of telemetry data, you can upload the telemetry backup to a remote location. To upload a backup to a remote location, use the following command.

```
request paragon backup upload backup-id backupID storage-location remote-path user username password password
```

You can re-upload the data to the same remote location folder every time you choose to back up telemetry data to maintain incremental copies.

## Restore Using Paragon Shell

You can restore your Paragon Automation network configuration from a backup configuration folder. To restore from a backup configuration folder, all microservices and applications must be stopped, and the cluster is not functional until the databases are restored. Once the databases are flushed and restored to the backed-up configuration, the applications must be restarted, and configuration restored from the databases must be reparsed.

To restore your Paragon Automation configuration from a specific backup configuration folder.

1. Log in as root user to the node where the backup folder is located.
2. Execute the following command to uninstall and stop all running application services.

```
root@Primary1> request paragon service destroy
```

This command runs in the background and takes some time to complete. You must wait until all the applications are shut down before proceeding to the next step.

Monitor the progress of the command using the `monitor start /epic/config/log` command.

3. Clear S3 bucket.

- a. Type `exit` to exit to the Linux root shell.
- b. Clear S3 bucket.

```
root@Primary1:~# paragon-wipe-s3-bucket --bucket devicesoftware
```

- c. After the script completes, type `cli` to log in again to Paragon Shell.
4. List all the backup directories available.
  - If your backup directory is stored in the cluster nodes, use the `show paragon backup` command. Determine the location of the backup directory and log in to the cluster node as root user.

For example:

```
root@Primary1> show paragon backup
Establishing SSH connection with 10.1.2.6
Establishing SSH connection with 10.1.2.4
Establishing SSH connection with 10.1.2.5

Following successful backups are available:
```

Node	Backup ID	Backup Type
10.1.2.3	20250318-090214	all

- If your backup folder is stored in a remote location, view the backup folders, determine the backup ID and download the backup folder locally to the cluster node, using the following commands.

```
root@Primary1> show paragon backup remote storage-location remote-path user username password password
```

```
root@Primary1> request paragon backup download backup-id backupID storage-location remote-path user username password password
```

Where:

*remote-path* is the path to the remote backup folder.

*username* and *password* are the login credentials to the remote server.

5. Restore the applications configuration from the backup folder.

```
root@Primary1> request paragon restore backup-id backupID
```

You must perform the restore operation only on the node on which the required backup folder is located.

6. (Optional) View progress status of the restore job.

```
root@Primary1> show paragon restore status backup-id backupID
```

7. Reinstall all the application services.

```
root@Primary1> request paragon service start
```

Monitor the progress of the command using the `monitor start /epic/config/log` command.



**NOTE:** When you run the `request paragon service start` command, sometimes the command may fail because the **config.yml** is empty. Verify that the **config.yml** is empty using the `show /epic/config/config.yml` command. Perform the steps detailed in the [Release Notes: Installation and Upgrade](#) section to repopulate the **config.yml** file and reinstall the application services.

8. Ensure that all devices and links are visible in the topology map. Perform the following steps.

- a. Type `exit` to exit to the Linux root shell.

- b. Delete the `pf-org-id` namespace.

```
root@Primary1:~# kubectl delete namespace $(kubectl get namespaces -o jsonpath='{.items}' | jq -r '[] | select(.metadata.name | startswith("pf-"))|.metadata.name')
```

- c. Restart configmonitor.

```
root@Primary1:~# kubectl -n northstar rollout restart deployment ns-configmonitor
```

d. Type `cli` to log in again to Paragon Shell.

## 9. Reparse and synchronize the restored configuration.

```
root@Primary1> request paragon restore sync backup-id backupID
```

### Caveats of the restore process

- When you perform the restore operation, the network configuration is returned to the configuration present in the backup folder. From the time the backup was taken, if the network configuration has changed due to new devices being onboarded or new service orders being executed, the network configuration in Paragon Automation might be different from the actual network state. To ensure that the network configuration in Paragon Automation and the actual network state have minimal mismatch post a restore operation, we recommend that you take regular periodic backups or specific backups after every network intent change.
- You cannot restore data from a release different from the current installed release of Paragon Automation.
- Since a backup does not store the certificates and infrastructure services configurations, that information must be kept unchanged during restoration.
- Resources allocated to the network won't be preserved after a restore and you must ensure that you release the allocated resources during the window between taking a backup and performing a restore.
- Performing a restore operation requires a maintenance window. You must expect that all functionality, including access to the GUI, is unavailable during this time frame.

## View or Delete Backup Files

To view a list of all backup folders across all nodes, use the following command:

```
root@Primary1> show paragon backup
```

The node connects to all the other nodes in the Paragon Automation cluster using SSH and displays a list of all backup folder names along with the IP address of the node on which the folder is located.

To view a list of backup folders along with a list of failed backup attempts, use the following command:

```
root@Primary1> show paragon backup include-failure true
```

To list the available backups from a remote location use the following command.

```
root@Primary1> show paragon backup remote storage-location remote-path user username password password
```

Use this command to determine the backup that you want to download from the remote location. You can view the backup directories only in the folder that you have specified in the path and not all the backups available in other folders on the remote server. You cannot delete a backup in a remote location since you don't have the necessary permissions to manage the remote server.

To delete a backup folder, use the following command.

```
root@Primary1> request paragon backup delete backup-id backupID
```

You can delete a backup folder that is located only on the node on which you execute the command.

## Upload or Download Backup Files

Upload your backed-up folder to a remote location outside the Paragon Automation cluster, within the same network as the cluster or in a different network. To upload your backup folder to a remote location, use the following command:

```
root@Primary1> request paragon backup upload backup-id backupID storage-location scp://IP:port/remote-path user
username password password
```

To view progress of the backup folder upload command, use the following command:

```
root@Primary1> show paragon backup upload status backup-id backupID
```

To download your backup folder from a remote location, use the following command:

```
root@Primary1> request paragon backup download backup-id backupID storage-location scp://IP:port/remote-path user
username password password
```

To view progress of the backup folder download command, use the following command:

```
root@Primary1> show paragon backup upload status backup-id backupID
```

## RELATED DOCUMENTATION

[Upgrade Paragon Automation | 62](#)

*request paragon backup*

*request paragon restore*

*show paragon backup*

# Disaster Recovery using VMware ESXi

## IN THIS SECTION

- [Disaster Recovery | 110](#)
- [Back Up and Restore Using VMware Snapshots | 111](#)

This topic describes the backup and restore and disaster recovery functionality available for Paragon Automation on the VMware ESXi hypervisor.

## Disaster Recovery

You can restore data on a fresh installation from a backup take on a different setup using the procedure described in this section. You can back up your Paragon Automation cluster and all associated application configuration by exporting it to your ESXi server. Once exported, you can redeploy it on a new installation. During backup and re-installation, your cluster is shutdown and is nonoperational. Perform the following steps to export and redeploy:

1. Verify that the cluster is in good health. Log in to one of the primary nodes and execute the request `paragon health-check` command from the Linux root shell.

The Overall Cluster Status must be GREEN.

2. Export your current cluster configuration.

- a. Log in to one of the node VMs. You are placed in Paragon Shell.
- b. Shut down the cluster using the request `paragon shutdown type cluster` command.
- c. Log in to your ESXi server.
- d. Export the configuration of all VMs one after another and enter appropriate names when prompted.

Perform the steps described in the VMware vSphere documentation at [Export an OVF Template](#).

The VM configuration is backed up locally as four sets of OVF, `.vmdk`, `.nvram`, and `.mf` configuration files.

3. Redeploy the exported configuration as a fresh installation.

Perform the steps described in ["Create the Node VMs" on page 27](#). Select the corresponding sets of OVF and configuration files for each VM.

The cluster is re-installed afresh.

4. (Optional) Verify that the cluster is in good health. Log in to one of the cluster nodes and execute the request `paragon health-check` command.

The Overall Cluster Status must be GREEN.

## Back Up and Restore Using VMware Snapshots

### IN THIS SECTION

- [Back Up Using VMware Snapshots | 111](#)
- [Restore Using VMware Snapshots | 112](#)

You can use the snapshot functionality available on VMware vSphere and ESXi clients to back up and restore your Paragon Automation cluster and application configuration data. The backup procedure using the snapshot functionality can be performed while the microservices and applications are running and does not affect the operation of the network. The snapshot creates a copy of your cluster and application configuration information. The snapshot restore functionality enables you to restore your cluster on the same set of VMs from which the backup was taken. The VMs are suspended before performing a restore operation and the downtime associated with a restore operation is minimal.

### Back Up Using VMware Snapshots

To back up your Paragon Automation cluster and application configuration data, use any one of the following options:

- **On VMware vSphere client**

1. Log in to any of the cluster nodes.
2. Use the request `paragon health-check` Paragon Shell command to ensure that your cluster is in good health. The status of the cluster must be GREEN.
3. Log in to the VMware vSphere client which manages the Paragon Automation VMs.
4. Click on your cluster host in the VM inventory.

Your cluster node VMs are listed in the **Virtual Machines** tab under VMs.

5. Select all the VMs, right-click, and click **Take snapshot**. The snapshot creation dialog box appears.

6. Enter a name for the snapshot and a description. Also, select the **Include virtual machine's memory** check-box.
7. Click **Create** to create the snapshot. The snapshot creation takes a few minutes.

You can check the progress of the job under **Recent Tasks** or in the **Task Console**.

- **On VMware ESXi client**

1. Log in to any of the cluster nodes.
2. Use the request paragon health-check Paragon Shell command to ensure that your cluster is in good health. The status of the cluster must be GREEN.
3. Log in to the VMware ESXi server on which your Paragon Automation VMs are located.
4. Select a cluster node VM listed under **Virtual Machines**, right-click and click **Take snapshot**. The snapshot creation dialog box appears.
5. Enter a name for the snapshot and a description. Also, select the **Include virtual machine's memory** check-box.
6. Click **Create** to create the snapshot.
7. Repeat steps "4" on page 112 through "6" on page 112 for the other three VMs and enter appropriate snapshot names when prompted.



**NOTE:** We recommend that you create snapshots of all the VMs one after the other as close as possible to each other in time.

8. (Optional) You can check the progress of the snapshot jobs under **Recent Tasks** or in the **Task Console**.

- **Using APIs**

You can also use the `snapshot.create` option to take snapshots of VMs using APIs. For more information, see <https://github.com/vmware/govmomi/blob/main/govc/USAGE.md#snapshotcreate>.

## Restore Using VMware Snapshots

To restore your Paragon Automation cluster and application configuration data which was previously backed up in snapshots, use any one of the following options:

- **On VMware vSphere client**

1. Log in to the VMware vSphere client which manages the Paragon Automation VMs and where the snapshots were previously taken.
2. Click your cluster host in the VM inventory.

Your cluster VMs are listed in the **Virtual Machines** tab under VMs.

3. Select all the VMs, right-click, and click **Suspend** to suspend the operation of the VMs. Do not restore the VMs when they are powered on.
4. Select a cluster node VM, right-click, and click **Manage snapshots**. The Manage snapshots page appears with a list of available snapshots.
5. Select the snapshot that you want to restore and click **Restore snapshot**.

Ensure that you select a snapshot that is recent and stable.

6. Repeat steps "4" on page 113 through "5" on page 113 for the other three VMs.

Alternatively, you can select a VM, right-click and click **Revert to Latest Snapshot** if your immediate last snapshot was the most stable version.



**NOTE:** We recommend that you restore snapshots of all the VMs one after the other as close as possible to each other in time. Do not wait for the snapshot restoration of any one VM to complete before restoring the other.

7. (Optional) You can check the progress of the job under **Recent Tasks** or in the **Task Console**.
8. Once the restore process is complete, wait for a few minutes for the VMs to power-on and the cluster to stabilize.
9. Log in to any of the cluster nodes.
10. Type the `show paragon cluster nodes` command in Paragon Shell. Ensure that the status of all the nodes is Ready.
11. Type the `show paragon cluster pods` command and ensure that the status of all the pods is either Running or Completed.
12. Type the `request paragon health-check` command to ensure that your cluster is in good health. The status of the cluster must be GREEN.



**NOTE:** If the cluster health-check does not return a GREEN status, reboot all the nodes using the `request paragon reboot type cluster` command and recheck the status.

- **On VMware ESXi client**

1. Log in to the VMware ESXi server on which your Paragon Automation VMs are located.

2. Select your cluster node VM listed under **Virtual Machines**, right-click and click **Suspend** to suspend the operation of the VM. Do not restore a VM when it is powered on.
3. Right-click the selected VM and click **Manage snapshots**. The Manage snapshots page appears with a list of available snapshots.
4. Select the snapshot that you want to restore and click **Restore snapshot**.  
Ensure that you select a snapshot that is recent and stable.
5. Repeat steps "2" on page 114 through "4" on page 114 for the other three VMs.

Alternatively, you can select a VM, right-click and click **Revert to Latest Snapshot** if your immediate last snapshot was the most stable version.



**NOTE:** We recommend that you restore snapshots of all the VMs one after the other as close as possible to each other in time. Do not wait for the snapshot restoration of any one VM to complete before restoring the other.

6. (Optional) You can check the progress of the restore jobs under **Recent Tasks** or in the **Task Console**.
  7. Once the restore process is complete, wait for a few minutes for the VMs to power-on and the cluster to stabilize.
  8. Log in to any of the cluster nodes.
  9. Type the `show paragon cluster nodes` command in Paragon Shell. Ensure that the status of all the nodes is Ready.
  10. Type the `show paragon cluster pods` command and ensure that the status of all the pods is either Running or Completed.
  11. Type the `request paragon health-check` command to ensure that your cluster is in good health. The status of the cluster must be GREEN.
- **Using APIs**  
You can also use the `snapshot.revert` option to restore backed-up snapshots of VMs using APIs. For more information, see <https://github.com/vmware/govmomi/blob/main/govc/USAGE.md#snapshotrevert>.

## RELATED DOCUMENTATION