

Juniper Paragon Automation 2.4.0 Monitoring and Troubleshooting Guide

Published
2025-07-22

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 Monitoring and Troubleshooting 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 | v](#)

1

Monitoring

[Monitor Using Sources and Sinks | 2](#)

[Monitoring Overview | 2](#)

[Configure Sources and Sinks | 3](#)

[Sample Sources and Sinks Configuration | 4](#)

[Configure Monitoring | 12](#)

[Configure IBM QRadar as a Monitoring Sink | 13](#)

2

Troubleshooting

[Troubleshoot Using Paragon Shell | 17](#)

[Troubleshooting Overview | 17](#)

[Troubleshooting Commands | 18](#)

[Perform a Health Check | 24](#)

[Troubleshoot Using Linux Root Shell | 27](#)

[Check Storage Utilization | 27](#)

[Check PostgresDB | 33](#)

[Uploading and Deleting Software Images Fails | 35](#)

[Troubleshoot OpenSearch | 37](#)

[OpenSearch Debugging Commands | 37](#)

[Health Check | 37](#)

[List Indices | 38](#)

[Check Shards Status | 38](#)

[Increase OpenSearch Memory | 40](#)

[OpenSearch Fails During Cluster Upgrade | 42](#)

[Troubleshoot Rook and Ceph | 45](#)

About This Guide

Use this guide to monitor and troubleshoot Juniper Paragon Automation.

RELATED DOCUMENTATION

[Juniper Paragon Automation User Guide](#)

[Juniper Paragon Automation Installation and Upgrade Guide](#)

1

PART

Monitoring

- [Monitor Using Sources and Sinks](#) | 2
-

CHAPTER 1

Monitor Using Sources and Sinks

IN THIS CHAPTER

- [Monitoring Overview | 2](#)
- [Configure Sources and Sinks | 3](#)
- [Sample Sources and Sinks Configuration | 4](#)
- [Configure Monitoring | 12](#)
- [Configure IBM QRadar as a Monitoring Sink | 13](#)

Monitoring Overview

IN THIS SECTION

- [Benefits | 2](#)

Paragon Automation uses a CLI-based infrastructure to monitor cluster resource usage and logs in real-time. The monitoring process collects metrics from diverse sources and forwards the collected data to designated sinks (such as Prometheus and VictoriaMetrics).

Paragon Automation uses Vector.dev, an advanced observability pipeline platform, that streamlines the intricate process of collecting, transforming, and storing observability data. You can view this data to interpret and analyze performance metrics ensuring timely responses to potential issues.

Benefits

- Provides real-time data collection enabling visibility into resource usage and logs.
- Provides insights into performance of the Paragon Automation cluster, enabling prompt identification and resolution of potential issues.

- Improves system performance and reliability by enabling timely responses based on real-time performance metrics and log data.

Configure Sources and Sinks

IN THIS SECTION

- [General Configuration Overview | 3](#)

To set up monitoring, you must configure Paragon Automation to specify sources and destinations for the metrics. The following terms are used extensively in this topic:

- **Source**—A source is the type and origin from which the observability data is collected.
- **Sink**—A sink is the destination to which the collected and transformed data is sent. This data can be visualized, interpreted, and analyzed to gain insights into cluster resource usage and performance, service status, and logs.

General Configuration Overview

Configure sources and sinks from the Paragon Shell CLI configuration mode. To enter configuration mode, type `configure` in Paragon Shell.

```
root@node> configure
Entering configuration mode
[edit]
root@node#
```

To configure a source, use the following command:

```
user@node# set paragon monitoring source source_id scope source_type
```

Where:

- *source_id*—Used to index the source and specified by the user.
- *scope* —Specifies whether the metrics must be collected at the cluster level or at the node level.

- *source_type*—See ["Supported Node Sources" on page 5](#) for a list of supported source types.

Similarly, to configure a sink, use the following command:

```
user@node# set paragon monitoring sink sink_id sink_type
```

Where:

- *sink_id*—Used to index the sink and specified by the user.
- *source_type*—See ["Supported Sinks" on page 10](#) for a list of supported sink types.

To configure a sink to receive data from one or more sources, use the `inputs` keyword.

```
user@node# set paragon monitoring sink sink_id inputs (source_id|list of source_ids)
```

Where, the input entries must match the defined *source_ids*.

Majority of sources and sinks have their own options. Type `?` after *source_type* or *sink_type* to view all available options.

```
user@node# set paragon monitoring source source_id scope source_type ?
```

or

```
user@node# set paragon monitoring sink sink_id sink_type ?
```

Vector.dev, by default, supports a reservoir of different types of sources and sinks. Currently, a few major sources are integrated into Paragon Shell CLI. Refer to ["Sample Sources and Sinks Configuration" on page 4](#) for detailed configuration for each currently supported sources and sinks.

Sample Sources and Sinks Configuration

IN THIS SECTION

- [Supported Node Sources | 5](#)
- [Supported Cluster Sources | 8](#)

- Supported Sinks | 10
- Default Sources and Sinks | 11

Use the information provided in this topic to view sample source and corresponding sink configuration.

According to the *scope* and *format*, all sources can be categorized in the following ways:

- *cluster* or *node*—Specifies whether the scope of the collected data is cluster level or node level.
- *metric* or *log*—Specifies the format of the observability data that is collected.

Supported Node Sources

Paragon Automation supports the following node log and metric sources.

Syslog

Collect system logs from all primary and worker nodes within the Paragon Automation cluster.

Category: node, log

Sample source configuration:

```
root@node# set paragon monitoring source syslog node syslog
```

Sample sink configuration:

```
root@node# set paragon monitoring sink syslogvlog inputs syslog
root@node# set paragon monitoring sink syslogvlog elasticsearch mode bulk
root@node# set paragon monitoring sink syslogvlog elasticsearch healthcheck enabled false
root@node# set paragon monitoring sink syslogvlog elasticsearch api_version v8
root@node# set paragon monitoring sink syslogvlog elasticsearch compression gzip
root@node# set paragon monitoring sink syslogvlog elasticsearch endpoints http://
monitoring_node:9428/insert/elasticsearch/
root@node# set paragon monitoring sink syslogvlog elasticsearch query "X-Powered-
By#Vector#_msg_field#message#_time#timestamp#_stream_fields#appname,hostname,facility,procid,seve
rity,source_type"
```

Docker Log

Collect logs from all the docker containers in the primary and worker nodes within the Paragon Automation Kubernetes cluster.

Category: node, log

Sample source configuration:

```
root@node# set paragon monitoring source docker node docker_logs
(optional) root@node# set paragon monitoring source docker node docker_logs include_containers
container_id_or_name
(optional) root@node# set paragon monitoring source docker node docker_logs exclude_containers
container_id_or_name
```

Sample sink configuration:

```
root@node# set paragon monitoring sink dockervlog inputs docker
root@node# set paragon monitoring sink dockervlog elasticsearch mode bulk
root@node# set paragon monitoring sink dockervlog elasticsearch healthcheck enabled false
root@node# set paragon monitoring sink dockervlog elasticsearch api_version v8 set paragon
monitoring sink dockervlog elasticsearch compression gzip
root@node# set paragon monitoring sink dockervlog elasticsearch endpoints http://
monitoring_node:9428/insert/elasticsearch/
root@node# set paragon monitoring sink dockervlog elasticsearch query "X-Powered-
By#Vector#_msg_field#message#_time#timestamp#_stream_fields#container_name,container_id,stream,im
age"
```



NOTE: An implicit transform audit-parser is used internally and is required for this source. The source ID must be audit and the input field for the corresponding sink must be audit-parser.

Paragon Shell cMGD Log

Collect the cMGD log from Paragon Shell.

Category: node, log

Sample source configuration:

```
root@node# set paragon monitoring source cmgd node cmgd_log
```

Sample sink configuration:

```
root@node# set paragon monitoring sink cmgdvlog inputs cmgd
root@node# set paragon monitoring sink cmgdvlog elasticsearch mode bulk
root@node# set paragon monitoring sink cmgdvlog elasticsearch healthcheck enabled false
root@node# set paragon monitoring sink cmgdvlog elasticsearch api_version v8 set
root@node# paragon monitoring sink cmgdvlog elasticsearch compression gzip
set paragon monitoring sink cmgdvlog elasticsearch endpoints http://monitoring_node:9428/insert/
elasticsearch
root@node# set paragon monitoring sink cmgdvlog elasticsearch query "X-Powered-
By#Vector#_msg_field#message#_tim
```

Host Metric

Collect host resource usage from the Paragon Automation cluster nodes.

Category: node, metric

Sample source configuration:

```
root@node# set paragon monitoring source host node host_metrics scrape_interval_secs 60
```

Sample sink configuration:

```
root@node# set paragon monitoring sink vm inputs add-hostname
root@node# set paragon monitoring sink vm prometheus_remote_write endpoint http://
monitoring_node:8428/api/v1/write
root@node# set paragon monitoring sink vm prometheus_remote_write compression zstd
root@node# set paragon monitoring sink vm prometheus_remote_write healthcheck enabled false
```



NOTE: An implicit transform `add-hostname` is used internally to add the `hostname` field to the processed data. The source ID must be `host` and the input field for the corresponding sink must be `add-hostname`.

Supported Cluster Sources

Paragon Automation supports the following cluster log and metric sources.

Kubernetes Log

Collect logs from all Kubernetes pods.

Category: cluster, log

Sample source configuration:

```
root@node# set paragon monitoring source k8s cluster kubernetes_logs
```

Sample sink configuration:

```
root@node# set paragon monitoring sink kuberneteslog inputs k8s
root@node# set paragon monitoring sink kuberneteslog elasticsearch mode bulk
root@node# set paragon monitoring sink kuberneteslog elasticsearch healthcheck enabled false
root@node# set paragon monitoring sink kuberneteslog elasticsearch api_version v8
root@node# set paragon monitoring sink kuberneteslog elasticsearch compression gzip
root@node# set paragon monitoring sink kuberneteslog elasticsearch endpoints http://
monitoring_node:9428/insert/elasticsearch/
root@node# set paragon monitoring sink kuberneteslog elasticsearch query "X-Powered-
By#Vector#_msg_field#message#_time#timestamp#_stream_fields#kubernetes.pod_namespace,kubernetes.p
od_name,kubernetes.pod_node_name"
```

Audit Log

Collect logs from the Paragon Automation audit log.

Category: cluster, log

Sample source configuration:

```
root@node# set paragon monitoring source audit cluster kafka bootstrap_servers kafka.common:9092
root@node# set paragon monitoring source audit cluster kafka group_id vector-kafka-consumer
root@node# set paragon monitoring source audit cluster kafka topics audits-dev
```

Sample sink configuration:

```
root@node# set paragon monitoring sink auditvlog inputs audit-parser
root@node# set paragon monitoring sink auditvlog elasticsearch mode bulk
root@node# set paragon monitoring sink auditvlog elasticsearch healthcheck enabled false
root@node# set paragon monitoring sink auditvlog elasticsearch api_version v8
root@node# set paragon monitoring sink auditvlog elasticsearch compression gzip
root@node# set paragon monitoring sink auditvlog elasticsearch endpoints http://
monitoring_node:9428/insert/elasticsearch
root@node# set paragon monitoring sink auditvlog elasticsearch query "X-Powered-
By#Vector#_msg_field#message#_time#timestamp#_stream_fields#org_id,site_id,admin_name,src_ip"
```

Kube State Metric

Collect Kubernetes resource usage from kube-state-metric.

Category: cluster, metric

Sample source configuration:

```
root@node# set paragon monitoring source ksm cluster prometheus_scrape endpoints http://kube-
state-metrics.kube-system:8080/metrics
root@node# set paragon monitoring source ksm cluster prometheus_scrape scrape_interval_secs 60
```

Sample sink configuration:

```
root@node# set paragon monitoring sink vm inputs add-hostname
root@node# set paragon monitoring sink vm prometheus_remote_write endpoint http://
monitoring_node:8428/api/v1/write
root@node# set paragon monitoring sink vm prometheus_remote_write compression zstd
root@node# set paragon monitoring sink vm prometheus_remote_write healthcheck enabled false
```



NOTE: An implicit transform `add-hostname` is used internally to add the hostname field to the processed data. The source ID must be `ksm` and the input field for the corresponding sink must be `add-hostname`.

Kubernetes Container Metric

Collect container resource usage of Kubernetes pods in the Paragon Automation cluster.

Category: cluster, metric

Sample source configuration:

```
root@node# set paragon monitoring source k8s_container_metric cluster
kubernetes_container_metrics
```

Sample sink configuration:

```
root@node# set paragon monitoring sink cadvisor inputs k8s_container_metric
root@node# set paragon monitoring sink cadvisor prometheus_remote_write endpoint http://
monitoring_node:8428/api/v1/write
root@node# set paragon monitoring sink cadvisor prometheus_remote_write compression zstd
root@node# set paragon monitoring sink cadvisor prometheus_remote_write healthcheck enabled false
```

Supported Sinks

All sinks can also be categorized in the following way as `log` or `metric` to identify the format of the observability data that the sink accepts.

A data sink can accept input only from log sources and a metric sink can accept input only from metric sources.

Paragon Automation supports the following cluster log and metric sinks.

Elasticsearch

Send data to a destination that supports the Elasticsearch format.

Category: log

The available options are:

```
root@node# set paragon monitoring sink id elasticsearch ?
Possible completions:
  api_version      The API version of Elasticsearch
+ apply-groups     Groups from which to inherit configuration data
+ apply-groups-except Don't inherit configuration data from these groups
  compression      Data compression method. Default is none
+ endpoints        HTTP(S) endpoint of sources/sinks
> healthcheck      Whether or not to check the health of the sink when Vector starts up
  mode             Elasticsearch Indexing mode
```

`query` Custom parameters to add to the query string for each HTTP request sent to Elasticsearch. In the format of `arg1_key#arg1_value#arg2_key#arg2_value...`. Number of hashtag separated items has to be an even number

Prometheus Remote Write

Deliver metric data to a Prometheus remote write endpoint.

Category: metric

The available options are:

```
root@node# set paragon monitoring sink id prometheus_remote_write ?
Possible completions:
+ apply-groups          Groups from which to inherit configuration data
+ apply-groups-except  Don't inherit configuration data from these groups
  compression          Data compression method. Default is snappy
  endpoint              HTTP(S) endpoint
> healthcheck           Whether or not to check the health of the sink when Vector starts up
```

For more information, see https://prometheus.io/docs/practices/remote_write/.

Default Sources and Sinks

When the Paragon Automation cluster is installed for the first time, the following three sources are automatically created:

- Kube State Metric—`ksm`
- Host—`host`
- Audit log—`audit`

```
root@node# show paragon monitoring source ?
Possible completions:
<id>          ID of the source. Should be of pattern [a-z][a-z0-9_-]*
audit         ID of the source. Should be of pattern [a-z][a-z0-9_-]*
host          ID of the source. Should be of pattern [a-z][a-z0-9_-]*
ksm           ID of the source. Should be of pattern [a-z][a-z0-9_-]*
```

You can modify the configuration for each default source but the source must not be removed.

You must set up and configure your own sinks on your own network which the Paragon Automation cluster can access.

Configure Monitoring

Use the steps detailed in this topic to configure monitoring in Paragon Automation to collect metrics from different types of sources and forward the collected data to designated sinks.

1. Log in to the node from which you deployed the Paragon Automation cluster.
2. Type `configure` to enter configuration mode.
3. Configure the sources and sinks. Use the following commands.

- `root@primary1# set paragon monitoring source source_id scope source_type`

- `root@primary1# set paragon monitoring sink sink_id sink_type`

Where:

source_id and *sink_id* is the required source or sink ID.

Scope is cluster or node.

To view a list of all available *sink-options* and *source-options* as well as sample configurations, see *set paragon monitoring* and ["Sample Sources and Sinks Configuration" on page 4](#).

4. Type `commit` and `quit` to commit the configuration and exit the configuration mode.
Committing will update the monitoring configuration, but will not deploy the changes to the underlying services.
5. Deploy the monitoring updates.

```
root@primary1> request paragon deploy monitoring
Getting vector daemonset metadata...
Loading vector sources and sinks...
Validating config...
Deleting existing vector configmap...
Creating new vector configmap...
configmap/vector-config created
Vector source or sinks missing...
Suppressing vector pods
```

The vector pods are only spawned when at least one source and one sink is configured.

6. Verify that the vector pods are up and operational.

```
root@primary1> show paragon cluster pods namespace kube-system | grep vector
vector-jrsh2                                1/1    Running    0
44h
vector-lnlfl                                1/1    Running    0
44h
vector-pcndg                                1/1    Running    0
44h
vector-rnjnc                                1/1    Running    0
44h
```

7. Verify that data is received at the configured sink.

Configure IBM QRadar as a Monitoring Sink

IN THIS SECTION

- [Host Syslog | 13](#)
- [Other Logs Supported by Paragon Automation | 14](#)

You can configure Paragon Automation to send all types of log data to IBM QRadar. We recommend two approaches for different types of logs:

Host Syslog

System logs on Paragon Automation clusters are managed by rsyslog, which supports multiple output modules. Although Paragon Automation monitoring does support collecting these host system logs, you can configure rsyslog to directly forward the system log to QRadar.

To configure rsyslog to send system log-data to QRadar:

1. Log in to a Paragon Automation cluster node and type `exit` to access the Linux root shell.
2. Navigate to the `/etc/rsyslog.d/` directory.

3. Create a **.conf** configuration file using the rsyslog naming convention, or modify an existing configuration file.
4. Add the following line to the configuration file.

```
*. * action(type="omfwd" target="qradar_host" port="514" protocol="tcp" resumeRetryCount="-1"
queue.type="LinkedList" queue.filename="Forward1" queue.saveOnShutdown="on")
```

Replace *qradar_host* with your QRadar host IP address or hostname.

5. Restart the rsyslogd process.

```
# service rsyslog restart
```

Host system logs will start streaming into QRadar.

Repeat this process on the remaining Paragon Automation cluster nodes.

Other Logs Supported by Paragon Automation

For all other types of logs (Kubernetes container log, Docker log, Audit log) supported by Paragon Automation monitoring, perform the following steps to send system data to QRadar.

1. Log in to a Paragon Automation cluster node and type `configure` in Paragon Shell to enter the configuration mode.
2. Enter the following commands in configuration mode.

```
root@node# set paragon monitoring sink qradar inputs ID
root@node# set paragon monitoring sink qradar socket address QRadar_IP_address:514
root@node# set paragon monitoring sink qradar socket mode tcp
root@node# set paragon monitoring sink qradar socket encoding codec raw_message
```

Replace *ID* with the ID of the log source. Retrieve the source ID using the `show paragon monitoring source ?` command.

To add multiple inputs, repeat the `inputs` command for different IDs or specify a list of inputs.

```
root@node# set paragon monitoring sink qradar inputs [k8s_log docker_log]
```

3. Type `commit` and `quit` to commit the configuration and exit configuration mode.

4. Deploy the monitoring updates.

```
root@node> request paragon deploy monitoring
```

2

PART

Troubleshooting

- [Troubleshoot Using Paragon Shell | 17](#)
 - [Troubleshoot Using Linux Root Shell | 27](#)
 - [Troubleshoot OpenSearch | 37](#)
 - [Troubleshoot Rook and Ceph | 45](#)
-

Troubleshoot Using Paragon Shell

IN THIS CHAPTER

- [Troubleshooting Overview | 17](#)
- [Troubleshooting Commands | 18](#)
- [Perform a Health Check | 24](#)

Troubleshooting Overview

IN THIS SECTION

- [Benefits | 18](#)

Paragon Automation enables you to troubleshoot and debug issues with your Paragon Automation deployment by using Paragon Shell CLI troubleshooting commands. These troubleshooting commands enhance your problem resolution capabilities with specific commands that gather support and troubleshooting information, enabling you to pinpoint and resolve cluster-related issues effectively. The commands enable the discovery of cluster-related issues by executing a series of support commands sequentially from any cluster node.

The Paragon Shell CLI troubleshooting commands are:

- [request paragon support information on page 19](#)—Provides an in-depth status report of your Paragon Automation cluster configuration. The command output includes information such as CPU and memory usage, information about pods, nodes, namespaces, persistent volume claim (PVC), persistent volume (PV), and so on.
- [request paragon troubleshooting information on page 21](#)—Provides troubleshooting logs of the `ems`, `foghorn`, `insights`, `paa`, `trust`, and `pathfinder` services.

When you run the troubleshooting command, a `troubleshooting_date_time.tar.gz` file is generated. Share this file with [Juniper Technical Assistance Center \(JTAC\)](#) for further evaluation. This `.tar.gz` file is saved in the `/root/troubleshooting/` directory.

Paragon Automation also provides robust debugging commands that gather data from key system components such as the Redis database, Kafka messages, service logs, time series database (TSDB), Helm and Kubernetes deployment service information, and so on. These commands are not part of the Paragon Shell CLI troubleshooting commands, and must be run separately. See "[Additional Troubleshooting Commands to Debug Issues](#)" on [page 22](#) for more information.

Use the troubleshooting commands when you see issues such as unavailable data across devices, syslog, BGP, IS-IS, or incorrect fan status, incorrect interface availability, and so on. Analyze the data collected from the output of these commands to find the cause of any issues seen in the Paragon Automation cluster.

Benefits

- Streamlines the troubleshooting process by allowing you to execute multiple commands sequentially from any cluster node.
- Provides comprehensive diagnostic information by collecting data from various sources such as Redis, Kafka, service logs, Postgres, and TSDB, enabling a thorough understanding of the deployment state.
- Provides you comprehensive data logs at one place thereby reducing the time and effort needed to diagnose issues.
- Enables you to investigate and resolve complex issues with detailed data collected from all critical system components.

Troubleshooting Commands

IN THIS SECTION

- [request paragon support information | 19](#)
- [request paragon troubleshooting information | 20](#)
- [Additional Debugging Commands | 22](#)

Use this topic to learn more about the Paragon Automation support and troubleshooting commands.

request paragon support information

The `request paragon support information` command displays an in-depth status report of your Paragon Automation cluster configuration.

The show commands that are executed when you run the `request paragon support information` command are listed in [Table 1 on page 19](#).

Table 1: request paragon support information Commands

Command	Description
<code>show paragon cluster nodes</code>	Shows node information of your Paragon Automation cluster.
<code>show paragon cluster pods</code>	Shows pod information of your Paragon Automation cluster.
<code>show paragon cluster namespaces</code>	Shows namespace information of your Paragon Automation cluster.
<code>show paragon cluster details</code>	Shows storage and controller node information of your Paragon Automation cluster.
<code>show paragon version</code>	Shows the version of your Paragon Automation cluster.
<code>show paragon images version</code>	Shows the version of pods in your Paragon Automation cluster.
<code>show paragon cluster pods namespace healthbot sort memory</code>	Shows the top pods of the healthbot namespace sorted by memory utilization.
<code>show paragon cluster pods namespace healthbot sort cpu</code>	Shows the top pods of the healthbot namespace sorted by CPU utilization.

Table 1: request paragon support information Commands (Continued)

Command	Description
show paragon pvc details	Shows the persistent volume (PV) and persistent volume claim (PVC) information.

The request paragon support information command also runs many kubectl commands. These commands provide you debugging information such as Helm deployment service information for the NorthStar namespace, Kubernetes deployment information for api-aggregator service, and so on.

request paragon troubleshooting information

The request paragon troubleshooting information command provides troubleshooting information of the ems, foghorn, insights, paa, trust, and pathfinder Paragon Automation services.

To view the list of available services, run the following command:

```
user@node> request paragon troubleshooting information service ?
```

Possible completions:

ems	ems service
foghorn	foghorn service
insights	insights service
paa	paa service
pathfinder	pathfinder service
trust	trust service

When you run the request paragon troubleshooting information command, a troubleshooting_<date_time>.tar.gz file is generated. You can share this file with the [Juniper Technical Assistance Center \(JTAC\)](#) for further evaluation. This .tar.gz file is saved in the /root/troubleshooting/ directory.

The commands that are executed when you run the request paragon troubleshooting information command are listed in [Table 2 on page 21](#).

Table 2: request paragon troubleshooting information Commands

Command	Description
request paragon debug logs namespace healthbot service <i>service-name</i>	Generates log files of different services within the healthbot namespace. Replace <i>service-name</i> with: • tsdb-shim • tand • jtimon • config-server • api-server • analytical-engine • alerta
request paragon debug logs namespace foghorn service <i>service-name</i>	Generates log files of different services within the foghorn namespace. Replace <i>service-name</i> with: • order-management • placement • cmgd
request paragon debug logs namespace airflow service <i>service-name</i>	Generates a log file of the workflow-manager service within the airflow namespace.
request paragon debug logs namespace northstar service <i>service-name</i>	Generates log files of different services within the northstar namespace. Replace <i>service-name</i> with: • toposerver • web • configmonitor • api-aggregator

Table 2: request paragon troubleshooting information Commands (Continued)

Command	Description
<code>request paragon debug logs namespace papi service <i>service-name</i></code>	Generates log files of different services within the papi namespace. Replace <i>service-name</i> with: • oc-term • papi • papi-ws
<code>request paragon debug postgres</code>	Generates a text file (JSON format) with Postgres information.

Additional Debugging Commands

You can run commands to collect data from the Redis database, Kafka messages, service logs, and the time series database (TSDB). You can use this data to troubleshoot issues with your Paragon Automation cluster. These commands are not part of the Paragon Shell CLI troubleshooting commands, and must be run separately. [Table 3 on page 22](#) lists the commands.

Table 3: Additional Commands to Debug Issues

Command	Description
<i>Kafka</i>	
<code>request paragon debug kafka ?</code>	Display possible completions for the request paragon debug kafka command.
<code>request paragon debug kafka options "-C -t <i>topic-name</i> -o s@<i>start-time</i> -o e@<i>end-time</i> -e -JB" output-file "<i>file-name</i>"</code>	Generate an output file of Kafka messages for a topic for a specified period of time.
<i>Insights Kafka</i>	

Table 3: Additional Commands to Debug Issues (*Continued*)

Command	Description
request paragon debug insights-kafka-data ?	Display possible completions for the request paragon debug insights-kafka-data command.
request paragon debug insights-kafka-data device " <i>device-id</i> " time-period " <i>duration</i> "	Display insights-kafka-data information for a device, for a specific time period. An output file of the information is generated.
<i>Redis</i>	
request paragon debug redis ?	Display possible completions for the request paragon debug redis command.
request paragon debug redis redis-key-pattern "insights"	Display Redis key pattern information for redis-keys with pattern "insights".
request paragon debug redis-key-pattern "insights" output file	Generate an output file of Redis key pattern information for redis-keys with pattern "insights".
<i>Service Logs</i>	
request paragon debug logs ?	Display possible completions for the request paragon debug logs command.
request paragon debug logs namespace <i>name</i> service <i>service-name</i> time <i>duration</i>	Generate a log file for a service within a namespace for the specified time period.
<i>TSDB</i>	
request paragon debug get-tsdb-data ?	Display possible completions for the request paragon debug get-tsdb-data command.

Table 3: Additional Commands to Debug Issues *(Continued)*

Command	Description
<code>request paragon debug get-tsdb-data device <i>device-id</i> topic "<i>topic-name</i>" output <i>file</i></code>	Generate an output file of TSDB data for a particular device.
<i>Postgres</i>	
<code>request paragon debug postgres ?</code>	Display possible completions for the request paragon debug postgres command.
<code>request paragon debug postgres database <i>database-name</i> username <i>username</i> measurement <i>measurement-name</i> output (file)</code>	Generate an output file of the measurement value information of the Postgres database.

Perform a Health Check

IN THIS SECTION

- Purpose | 24
- Action | 24
- Sample Output | 25
- Meaning | 25

Purpose

Perform a health check on the cluster and get an overall status of the cluster.

Action

Log in to a cluster node and use the `request paragon health-check` command in Paragon Shell.

Sample Output

```

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

```

Meaning

The command performs multiple health checks on the cluster and returns a detailed list of all the tests run and each of their results. The health-check command checks for multiple parameters such as:

- Kubernetes status
- Health of each node (CPU, disk space, memory, I/O latency, and so on)
- Database health (Postgres, ArangoDB, OpenSearch, Kafka, and so on)
- Ceph storage health

The overall health status is categorized as green, amber, or red. A green status indicates a healthy cluster and that all health checks have passed successfully. A red status indicates that many health checks have failed and implies serious issues in the cluster. An amber status indicates that there may be some noncritical issues in the cluster. The status is returned as amber in the following instances:

- Nodes have taints
- Disk usage or memory usage on any node exceeds 80% of available space
- Disk I/O latency on any node exceeds 100000 ms
- Rook Ceph status shows HEALTH_WARN
- Number of kafka in-sync replicas are not equal to the number of kafka replicas



NOTE: Alternatively, you can also use `health-check` command from the Linux root shell to get an overall status of the cluster.

CHAPTER 3

Troubleshoot Using Linux Root Shell

IN THIS CHAPTER

- [Check Storage Utilization | 27](#)
- [Check PostgresDB | 33](#)
- [Uploading and Deleting Software Images Fails | 35](#)

Check Storage Utilization

IN THIS SECTION

- [Purpose | 27](#)
- [Action | 27](#)
- [Sample Output | 28](#)
- [Meaning | 32](#)

Purpose

Check utilization of local storage PVC, Ceph-based storage PVC, and the S3 bucket. When a health-check shows abnormal disk pressure or pods crash due to disk issues, you can check storage utilization to pinpoint the issue.

Action

Log in to the Linux root shell of a cluster node and use the following commands:

- For the local storage PVC—`paragon-local-volume-check`
- For Ceph-based storage PVC and S3 bucket—`paragon-ceph-usage-check [-h] [-s] [-c] [-b]`

Where:

- -h or --help displays command usage information.
- -c or --cephfs checks Ceph-based storage usage.
- -s or --s3 checks S3 bucket usage.
- -b or --both checks both Ceph-based storage and S3 bucket usage.

Sample Output

paragon-local-volume-check

```
root@pa1:~# paragon-local-volume-check
```

```
=====
```

```
Get local volume usage.
```

```
=====
```

pv-name	pvc-claim	local-path	disk-usage	node-name
echo	local-pv-36c71c79	foghorn-dbserver-6nrhq4nc		/export/local-
volumes/pv16	107M	pa1		
local-pv-49d4aff9		data-zookeeper-0		/export/local-
volumes/pv3	996K	pa1		
local-pv-aa9c4410		data-kafka-2		/export/local-
volumes/pv11	142M	pa1		
local-pv-c747791f		vmstorage-db-vmstorage-victoria-metrics-cluster-2		/export/local-
volumes/pv9	68K	pa1		
local-pv-ead7ada4		opensearch-cluster-master-opensearch-cluster-master-2		/export/local-
volumes/pv20	1.5M	pa1		
local-pv-22f5300c		pgdata-atom-db-2		/export/local-
volumes/pv5	519M	pa2		
local-pv-37c26b76		foghorn-dbserver-9d9nd112		/export/local-
volumes/pv3	106M	pa2		
local-pv-53c4d5b		foghorn-agent-hiknywjh		/export/local-
volumes/pv15	8.2M	pa2		
local-pv-80a32482		vmstorage-db-vmstorage-victoria-metrics-cluster-0		/export/local-
volumes/pv12	68K	pa2		
local-pv-92b5cf80		data-zookeeper-2		/export/local-
volumes/pv14	996K	pa2		
local-pv-a5c36300		vmstorage-db-vmstorage-victoria-metrics-cluster-0		/export/local-
volumes/pv1	140K	pa2		

local-pv-3abab0b0	pgdata-atom-db-1	/export/local-
volumes/pv17	519M pa3	
local-pv-4961bb7d	foghorn-agent-hpvuvjh9	/export/local-
volumes/pv5	8.2M pa3	
local-pv-6b46253c	opensearch-cluster-master-opensearch-cluster-master-0	/export/local-
volumes/pv13	2.3M pa3	
local-pv-7358834e	foghorn-dbserver-ddxwbr01	/export/local-
volumes/pv11	134M pa3	
local-pv-7e53b8bc	data-kafka-1	/export/local-
volumes/pv6	142M pa3	
local-pv-b788099	vmstorage-db-vmstorage-victoria-metrics-cluster-1	/export/local-
volumes/pv9	140K pa3	
local-pv-269c69f0	data-zookeeper-1	/export/local-
volumes/pv7	996K pa4	
local-pv-3d6bab16	vmstorage-db-vmstorage-victoria-metrics-cluster-2	/export/local-
volumes/pv10	72K pa4	
local-pv-40e3ef1	opensearch-cluster-master-opensearch-cluster-master-1	/export/local-
volumes/pv20	2.0M pa4	
local-pv-4eb12e61	pgdata-atom-db-0	/export/local-
volumes/pv15	581M pa4	
local-pv-866f53aa	data-kafka-0	/export/local-
volumes/pv9	142M pa4	
local-pv-abda8deb	vmstorage-db-vmstorage-victoria-metrics-cluster-1	/export/local-
volumes/pv17	68K pa4	
local-pv-c101544f	foghorn-agent-wldughyr	/export/local-
volumes/pv13	8.2M pa4	

paragon-ceph-usage-check -h

```

root@pal:~# paragon-ceph-usage-check -h
usage: paragon-ceph-usage-check [-h] [-s] [-c] [-b]

Helper command to check S3 and Ceph usage

options:
  -h, --help      show this help message and exit
  -s, --s3        Check S3 bucket usage
  -c, --cephfs    Check CephFS usage
  -b, --both      Check usage for both S3 and CephFS

```

paragon-ceph-usage-check -s

```

root@pal:~# paragon-ceph-usage-check -s
Checking S3 bucket usage
Listing top-level directories and checking usage:
S3 Directory list : ['devicesoftware', 'usw2-jcloud-dev-paa-plugin-service-plugins-storage']
Usage for: devicesoftware
Total Objects: 0
    Total Size: 0 Bytes
Usage for: usw2-jcloud-dev-paa-plugin-service-plugins-storage
Total Objects: 0
    Total Size: 0 Bytes

```

paragon-ceph-usage-check -c

```

root@pal:~# paragon-ceph-usage-check -c

Checking CephFS usage
PV to PVC , Volume and Size mapping
+-----+-----+-----+
+-----+-----+-----+
+-----+
|          PV          |          Claim          | Capacity
|          |          |          |
|          |          |          |
|          |          |          |
|          |          |          |
+-----+-----+-----+
+-----+-----+-----+
+-----+
| pvc-3eed5171-5efe-4c6b-8629-613844711362 | paa/timescaledb-data-paa-timescaledb-0 | 32Gi
| /volumes/csi/csi-vol-c96deb77-7145-403e-8da1-e971883ce6c1/0ba12eb7-e93d-47d4-afa1-21b8dc998971
| 0.0723346984013915 |
| pvc-42fec79a-fbf0-40bd-9dc5-066ff2f479e3 | common/redis-data | 10Gi
| /volumes/csi/csi-vol-0f8878fa-eeb5-4ce5-8396-2ee2766e9413/d8810464-bf81-4008-98e1-da295a579e35
| 2.360902726650238e-06 |
| pvc-46209d4a-2283-4fc2-aa55-2b9a4f9af6a1 | common/opensearch-backup | 32Gi
| /volumes/csi/csi-vol-9c5ace5e-d133-40aa-954d-fd51cceb2d80/3a8d0c37-4cfc-4ae7-8a82-a5deca1225b7
| 1.773890107870102e-05 |
| pvc-5ae0fd7f-b53d-4b78-ba7d-4b0297ed6e30 | healthbot/insights-data | 10Gi
| /volumes/csi/csi-vol-d51fa0b1-5432-4e6e-97b9-17d0df1bdf47/43435330-12b4-4a50-9ca7-efee19b94e96
| 0.0024269744753837585 |

```

```

| pvc-697325c2-4c93-4a1c-b60a-feb1d5b5b611 | license-client/license-client-data | 8Mi
| /volumes/csi/csi-vol-fc9ff7c8-e4a1-42f2-9f96-5c34f2cd559b/9f364508-9d15-4d13-8d83-c64da1bb7c25
| 0.0 |
| pvc-99cf22c0-33f3-44b1-ad53-2a898b3247b4 | common/opensearch-cephfs-pvc | 10Gi
| /volumes/csi/csi-vol-d3ba7888-09f3-45c6-8378-028f78d8b318/5b21acb7-6e02-428c-aacf-7ec78efd9932
| 0.0 |
| pvc-cdef057b-21ea-4481-9504-e8ff9f9094a0 | paa/orchestrator-volume | 10Gi
| /volumes/csi/csi-vol-f606b8f9-f705-498e-b0ee-f3afa4ed0d14/0ee7375d-908b-4230-a513-b6c6c872f73a
| 0.2804222572594881 |
| pvc-e160c14c-8832-4ad8-bd39-9ca43c1f1dca | airflow/airflow | 10Gi
| /volumes/csi/csi-vol-2b8e1186-650e-4489-b1cd-697c38511c01/a9c8d233-ea01-4d4d-9497-fa8f73b5875c
| 0.6222417075186968 |
+-----+-----+-----+
+-----+
+-----+
+-----+

```

paragon-ceph-usage-check -b

```

root@pal:~# paragon-ceph-usage-check -b
Checking S3 bucket usage
Listing top-level directories and checking usage:
S3 Directory list : ['devicesoftware', 'usw2-jcloud-dev-paa-plugin-service-plugins-storage']
Usage for: devicesoftware
Total Objects: 0
    Total Size: 0 Bytes
Usage for: usw2-jcloud-dev-paa-plugin-service-plugins-storage
Total Objects: 0
    Total Size: 0 Bytes

Checking CephFS usage
PV to PVC , Volume and Size mapping
+-----+-----+-----+
+-----+
+-----+
+-----+
|          PV          |          Claim          | Capacity
|          Volume location
|      Gb Used      |
+-----+-----+-----+
+-----+
+-----+
+-----+
| pvc-3eed5171-5efe-4c6b-8629-613844711362 | paa/timescaledb-data-paa-timescaledb-0 | 32Gi

```

```

| /volumes/csi/csi-vol-c96deb77-7145-403e-8da1-e971883ce6c1/0ba12eb7-e93d-47d4-afa1-21b8dc998971
| 0.0724263722077012 |
| pvc-42fec79a-fbf0-40bd-9dc5-066ff2f479e3 | common/redis-data | 10Gi
| /volumes/csi/csi-vol-0f8878fa-eeb5-4ce5-8396-2ee2766e9413/d8810464-bf81-4008-98e1-da295a579e35
| 2.360902726650238e-06 |
| pvc-46209d4a-2283-4fc2-aa55-2b9a4f9af6a1 | common/opensearch-backup | 32Gi
| /volumes/csi/csi-vol-9c5ace5e-d133-40aa-954d-fd51cceb2d80/3a8d0c37-4cfc-4ae7-8a82-a5deca1225b7
| 1.773890107870102e-05 |
| pvc-5ae0fd7f-b53d-4b78-ba7d-4b0297ed6e30 | healthbot/insights-data | 10Gi
| /volumes/csi/csi-vol-d51fa0b1-5432-4e6e-97b9-17d0df1bdf47/43435330-12b4-4a50-9ca7-efee19b94e96
| 0.0024269744753837585 |
| pvc-697325c2-4c93-4a1c-b60a-feb1d5b5b611 | license-client/license-client-data | 8Mi
| /volumes/csi/csi-vol-fc9ff7c8-e4a1-42f2-9f96-5c34f2cd559b/9f364508-9d15-4d13-8d83-c64da1bb7c25
| 0.0 |
| pvc-99cf22c0-33f3-44b1-ad53-2a898b3247b4 | common/opensearch-cephfs-pvc | 10Gi
| /volumes/csi/csi-vol-d3ba7888-09f3-45c6-8378-028f78d8b318/5b21acb7-6e02-428c-aacf-7ec78efd9932
| 0.0 |
| pvc-cdef057b-21ea-4481-9504-e8ff9f9094a0 | paa/orchestrator-volume | 10Gi
| /volumes/csi/csi-vol-f606b8f9-f705-498e-b0ee-f3afa4ed0d14/0ee7375d-908b-4230-a513-b6c6c872f73a
| 0.2804222572594881 |
| pvc-e160c14c-8832-4ad8-bd39-9ca43c1f1dca | airflow/airflow | 10Gi
| /volumes/csi/csi-vol-2b8e1186-650e-4489-b1cd-697c38511c01/a9c8d233-ea01-4d4d-9497-fa8f73b5875c
| 0.6234864285215735 |
+-----+-----+-----+
+-----+-----+-----+
+-----+

```

Meaning

The `paragon-local-volume-check` command lists all the PVCs, the local path, disk usage, and size of a PVC, as well as the node on which the PVC is located.

The `paragon-ceph-usage-check` command displays a detailed view of the Ceph filesystem with the usage, capacity, claim name, and PV name, as well as the volume location.

- Purpose | 33
- Action | 33
- Sample Output | 33
- Meaning | 34

Confirm the health of the PostgreSQL database by listing the states of the leaders and replicas. You must have, at a minimum, one leader in running state and two replicas in streaming state for PostgreSQL to be considered healthy. If any replica is in a non-streaming state (such as stopped, starting, and so on), you can reinitialize the replica.

Log in to an atom-db pod using the `kubectrl exec -it -n common atom-db-0 -- bash` command. List the cluster name, and all the replicas and leaders using the `patronictl list` command.

Sample Output

```

      _ _ _ _      _ _
/  _ _ | _ _ ( _ ) | _ _
\ _ _ \ | ' _ \ | | / _ \
  _ _ ) | | _ ) | | | ( _ ) |
| _ _ _ / | . _ _ / | _ _ | \ _ _ /
      | _ |

```

This container is managed by runit, when stopping/starting services use sv

Examples:

```
sv stop cron
sv restart patroni
```

Current status: (sv status /etc/service/*)

```
run: /etc/service/patroni: (pid 32) 589651s
```

```
run: /etc/service/pgqd: (pid 33) 589651s
```

```
root@atom-db-0:/home/postgres# patronictl list
```

```
+ Cluster: atom-db (7464654262273740859) -----+-----+-----+
| Member   | Host           | Role   | State   | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| atom-db-0 | 10.1.2.37      | Leader | running | 1  |           |
| atom-db-1 | 10.1.3.46      | Replica | streaming | 1  | 0         |
| atom-db-2 | 10.1.4.51      | Replica | streaming | 1  | 0         |
+-----+-----+-----+-----+-----+-----+
```

```
root@atom-db-0:/home/postgres# patronictl reinit atom-db atom-db-2
```

```
+ Cluster: atom-db (7464654262273740859) -----+-----+-----+
| Member   | Host           | Role   | State   | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| atom-db-0 | 10.1.2.37      | Leader | running | 1  |           |
| atom-db-1 | 10.1.3.46      | Replica | streaming | 1  | 0         |
| atom-db-2 | 10.1.4.51      | Replica | stopped  | 1  | 0         |
+-----+-----+-----+-----+-----+-----+
```

```
Are you sure you want to reinitialize members atom-db-2? [y/N]: y
```

```
Success: reinitialize for member atom-db-2
```

Where, atom-db is the cluster name, and atom-db-2 is the replica that you want to reinitialize.

Meaning

You must have at least one leader in the running state and two replicas in the streaming state for PostgreSQL to be considered healthy.

Uploading and Deleting Software Images Fails

IN THIS SECTION

- Problem | 35
- Cause | 35
- Solution | 35

Problem

You are unable to upload or delete software images from the Software Images page.

Cause

Ceph storage is full.

Solution

Increase the storage limit so that you can upload or delete software images from the Software Images page.

To modify the storage limit:

1. Log in to the primary node of your Kubernetes cluster.
2. Increase the storage limit.
 - a. Run the following command to edit the storage configuration:

```
kubect1 edit -n rook-ceph cephobjectstore object-store
```

- b. Navigate to **spec > dataPool > quotas** to modify the `maxSize` parameter.

```
spec:
  dataPool:
    quotas:
      maxSize: <new-size>
```

- c. Change the `maxSize` to the desired value. For example, you can increase the default storage value (16 Gi) to 24 Gi.
- d. Restart the service to push the new configuration by executing the following command:

```
kubectl rollout restart deployment rook-ceph-rgw-object-store-a -n rook-ceph
```



NOTE: You might experience a downtime of 2 to 3 minutes.

Troubleshoot OpenSearch

IN THIS CHAPTER

- [OpenSearch Debugging Commands | 37](#)
- [OpenSearch Fails During Cluster Upgrade | 42](#)

OpenSearch Debugging Commands

SUMMARY

This topic lists commonly used commands to debug and view OpenSearch status.

IN THIS SECTION

- [Health Check | 37](#)
- [List Indices | 38](#)
- [Check Shards Status | 38](#)
- [Increase OpenSearch Memory | 40](#)

Health Check

Use the `paragon-utils curl http://opensearch-cluster-master.common:9200/_cat/health?v` command to view the overall status of OpenSearch.

```
root@pal:~# paragon-utils curl http://opensearch-cluster-master.common:9200/_cat/health?v
epoch      timestamp cluster      status node.total node.data discovered_cluster_manager
shards pri  relo init unassign pending_tasks max_task_wait_time active_shards_percent
1738252967 16:02:47 opensearch-cluster green          3          3
true      54  23   0   0       0           0           -           100.0%
```

The database is considered healthy if the status is green. A green status indicates that all shards are assigned to nodes. A yellow status indicates that primary shards are assigned to nodes, but some replicas are not. A red status indicates that at the least one primary shard is not assigned to a node.

List Indices

Use the `paragon-utils curl -s http://opensearch-cluster-master.common:9200/_cat/indices?v` command to list all OpenSearch indices and their sizes.

```
root@pal:~# paragon-utils curl -s http://opensearch-cluster-master.common:9200/_cat/indices?v
health status index                                uuid                                pri rep docs.count
docs.deleted store.size pri.store.size
green open ask-paragon-chat-sessions-000001 yvDkfiCDSWS4HlvuSPN-Pg 3 2
0 0 1.8kb 624b
green open .plugins-ml-config WNRjb532QNqnTx3VMXrFFQ 1 1
1 0 7.8kb 3.9kb
green open .opensearch-observability g5NuXSgcR5WXMxkrh_Sazg 1 2
0 0 624b 208b
green open audits_00001 VQKBbdkKQTuQ0jcmXQ3DvQ 1 1
4 0 88.9kb 44.4kb
green open ask-paragon-chat-history-000001 J5kScLMgRfS9ZmGV3u0xPw 3 2
0 0 1.8kb 624b
green open .opendistro-job-scheduler-lock 7wu2FIqaSlqLEk6BT0SaJg 1 1
3 159 149.9kb 71.7kb
green open audits-202501 7ZAFD0QhTwyzKxWztEB_hQ 1 1
4 0 88.9kb 44.4kb
green open jcloud_alerts_active OfXRI1wtSQWJ8XRiXzU6-w 1 1
0 0 416b 208b
green open .ds-jcloud_alerts_cleared-000001 JMqwDN9bRe-uVap4Cml8uQ 6 1
0 0 2.4kb 1.2kb
```

Check Shards Status

Use the `paragon-utils curl http://opensearch-cluster-master:9200/_cat/shards?v` command to check OpenSearch primary and replica shards status, nodes to which they are assigned and their IP addresses.

```
root@pal:~# paragon-utils curl http://opensearch-cluster-master:9200/_cat/shards?v
index                                shard prirep state docs store
ip node
.opendistro-ism-managed-index-history-2025.01.30-000003 0 p STARTED
10.1.2.35 opensearch-cluster-master-1
.opendistro-ism-managed-index-history-2025.01.30-000003 0 r STARTED
10.1.2.49 opensearch-cluster-master-0
.opendistro-ism-config 0 p STARTED
10.1.2.35 opensearch-cluster-master-1
```

```

.opendistro-ism-config                                0    r    STARTED
10.1.2.49  opensearch-cluster-master-0
audits_00001                                          0    r    STARTED    4 44.4kb
10.1.2.53 opensearch-cluster-master-2
audits_00001                                          0    p    STARTED    4 44.4kb
10.1.2.49  opensearch-cluster-master-0
ask-paragon-chat-history-000001                     0    p    STARTED    0 208b
10.1.2.53 opensearch-cluster-master-2

<output snipped>

.ds-jcloud_alerts_cleared-000001                     4    p    STARTED    0 208b
10.1.2.35 opensearch-cluster-master-1
.ds-jcloud_alerts_cleared-000001                     4    r    STARTED    0 208b
10.1.2.49  opensearch-cluster-master-0
.ds-jcloud_alerts_cleared-000001                     5    r    STARTED    0 208b
10.1.2.35 opensearch-cluster-master-1
.ds-jcloud_alerts_cleared-000001                     5    p    STARTED    0 208b
10.1.2.49  opensearch-cluster-master-0

```

Consequently, if a shard status is not `STARTED`, use the `explain` API command to debug the reason.

```

root@pal:~# paragon-utils curl -s http://opensearch-cluster-master.common:9200/_cluster/
allocation/explain?pretty
{
  "index" : "ask-paragon-chat-history-000001",
  "shard" : 0,
  "primary" : false,
  "current_state" : "unassigned",
  "unassigned_info" : {
    "reason" : "NODE_LEFT",
    "at" : "2025-02-21T19:44:35.163Z",
    "details" : "node_left [T92oYspJRVSCyzycX3Am6A]",
    "last_allocation_status" : "no_attempt"
  },
  "can_allocate" : "no",
  "allocate_explanation" : "cannot allocate because allocation is not permitted to any of the
nodes",
  "node_allocation_decisions" : [
    {
      "node_id" : "6hSdCXELRDWmSQIy1pHBWg",
      "node_name" : "opensearch-cluster-master-0",

```

```

    "transport_address" : "10.1.2.3:9300",
    "node_attributes" : {
      "shard_indexing_pressure_enabled" : "true"
    },
    "node_decision" : "no",
    "deciders" : [
      {
        "decider" : "same_shard",
        "decision" : "NO",
        "explanation" : "a copy of this shard is already allocated to this node [[ask-paragon-
chat-history-000001][0], node[6hSdCXELRDWmSQIy1pHBWg], [R], s[STARTED],
a[id=G0v4Mf9WSNGJynneOBLQPQ]]"
      }
    ]
  },
  {
    "node_id" : "U2cN4Nv9TbCWL8J95crcdg",
    "node_name" : "opensearch-cluster-master-1",
    "transport_address" : "10.3.4.5:9300",
    "node_attributes" : {
      "shard_indexing_pressure_enabled" : "true"
    },
    "node_decision" : "no",
    "deciders" : [
      {
        "decider" : "same_shard",
        "decision" : "NO",
        "explanation" : "a copy of this shard is already allocated to this node [[ask-paragon-
chat-history-000001][0], node[U2cN4Nv9TbCWL8J95crcdg], [P], s[STARTED],
a[id=U1UPeU0IQHeDP9croBRngw]]"
      }
    ]
  }
]
}

```

Increase OpenSearch Memory

The value for Dynamic Memory Configuration for OpenSearch JVM options such as heap size setting is stored in `opensearch_java_opts` in the **config.yml** file. By default, the value is set to `opensearch_java_opts: "-Xmx4096M -Xms4096M"`. To increase OpenSearch memory:

1. Log in to a cluster node.

2. Type configure to enter configuration mode in Paragon Shell.
3. Use the following command to increase memory and enter the required memory value.

```
[edit]
root@pa1# set paragon cluster custom-config keyValue opensearch_java_opts string "Xmx5120M -
Xms5120M"

[edit]
```

4. Commit the configuration and exit configuration mode.

```
[edit]
root@pa1# commit
commit complete

[edit]
root@pa1# exit
```

5. Regenerate the configuration files.

```
root@pa1> request paragon config
Paragon inventory file saved at /epic/config/inventory
Paragon config file saved at /epic/config/config
```

6. Redeploy OpenSearch.

```
root@pa1> request paragon deploy cluster input "-t opensearch"
```

RELATED DOCUMENTATION

| [OpenSearch Fails During Cluster Upgrade](#) | 42

OpenSearch Fails During Cluster Upgrade

IN THIS SECTION

- Problem | 42
- Solution | 42

Problem

While upgrading Paragon Automation from an earlier release to the current release, OpenSearch might fail.

Solution

Restart Opensearch or restore OpenSearch data on the cluster node.

1. Log in to the Linux root shell of a cluster node and restart OpenSearch.

```
root@pa1:~# kubectl scale sts -n common opensearch-cluster-master --replicas=0; sleep 120;
kubectl scale sts -n common opensearch-cluster-master --replicas=3
statefulset.apps/opensearch-cluster-master scaled
statefulset.apps/opensearch-cluster-master scaled
```

2. Check OpenSearch health status using the `paragon-utils curl http://opensearch-cluster-master.common:9200/_cat/health?v` command. If the status is yellow or red, perform the following steps.
3. List all the pods.

```
root@pa1:~# kubectl get po -n common
```

NAME	READY	STATUS	RESTARTS	AGE
alert-manager-follower-5ccf865f99-7g7l5	1/1	Running	6 (4d1h ago)	6d23h
alert-manager-leader-659c4c888b-4lr9x	1/1	Running	6 (4d1h ago)	6d23h
atom-db-0	2/2	Running	0	6d23h
atom-db-1	2/2	Running	0	6d23h
atom-db-2	2/2	Running	0	6d23h
backup-server-644cc495dd-kxk26	1/1	Running	0	6d23h
backup-vmdb-01d7nh-gp9mk	0/1	Completed	0	5d8h

backup-vmdb-nqkop4-cdnfz	0/1	Completed	0	5d8h
backup-vmdb-zew5t9-prlbt	0/1	Completed	0	5d8h
cfssl-76474f7455-bwr7b	1/1	Running	0	6d23h
common-utils-865547d799-t2w78	1/1	Running	0	6d23h
kafka-0	2/2	Running	0	6d23h
kafka-1	2/2	Running	0	6d23h
kafka-2	2/2	Running	0	6d23h
local-volume-provisioner-6csjw	1/1	Running	0	7d
local-volume-provisioner-g6pzl	1/1	Running	0	7d
local-volume-provisioner-qnlh7	1/1	Running	0	7d
local-volume-provisioner-qsdxb	1/1	Running	0	7d
mailservice-9ff8dff9f-gfk5m	2/2	Running	0	6d23h
nats-0	3/3	Running	0	6d23h
nats-1	3/3	Running	0	6d23h
nats-2	3/3	Running	0	6d23h
nats-box-5886877c65-b486r	1/1	Running	0	6d23h
opensearch-backup-7f7dcbb77f-2q5p5	1/1	Running	0	6d23h
opensearch-backup-cron-28975680-x8wjp	0/1	Completed	0	17h
opensearch-backup-cron-28976040-z6p9m	0/1	Completed	0	11h
opensearch-backup-cron-28976400-9psp5	0/1	Completed	0	5h48m
opensearch-cleanup-cron-28972800-zqkkr	0/1	Completed	0	2d17h
opensearch-cleanup-cron-28974240-vr14k	0/1	Completed	0	41h
opensearch-cleanup-cron-28975680-2fppn	0/1	Completed	0	17h
opensearch-cluster-master-0	1/1	Running	0	4d1h
opensearch-cluster-master-1	1/1	Running	0	4d1h
opensearch-cluster-master-2	1/1	Running	0	4d1h
postgres-operator-78dcf4b786-96wdb	1/1	Running	0	6d23h
redis-master-5877db87fd-szvhp	1/1	Running	0	6d23h
zookeeper-0	1/1	Running	0	6d23h
zookeeper-1	1/1	Running	0	6d23h
zookeeper-2	1/1	Running	0	6d23h

4. Determine the name of the opensearch-backup-** pod and log in to the pod.

```
root@pa1:~# kubectl exec -it -n common opensearch-backup-7f7dcbb77f-2q5p5 -- bash
```

5. List the OpenSearch backup directories on the pod.

```
root@opensearch-backup-7f7dcbb77f-2q5p5:/# ls -l /opt/paragon/opensearch-backup
total 0
drwxr-xr-x 4 root root 2 Feb  2 18:00 20250202_180003-priority
```

```
drwxr-xr-x 4 root root 2 Feb  3 00:00 20250203_000003-priority
drwxr-xr-x 4 root root 2 Feb  3 06:00 20250203_060002-priority
drwxr-xr-x 4 root root 2 Feb  3 12:00 20250203_120003-priority
drwxr-xr-x 4 root root 2 Feb  3 18:00 20250203_180002-priority
drwxr-xr-x 3 root root 1 Feb  3 18:00 temp
```

6. Determine the backup directory that you want to restore and restore the backup.

```
root@opensearch-backup-7f7dcbb77f-2q5p5:/# common-utils/paragon-opensearch-restore /opt/  
paragon/opensearch-backup/20250203_180002-prioritydirectory  
2025-02-03 18:28:41 Usage: [backup_directory] [space-seperated indices_list]  
2025-02-03 18:28:41 Restoring OpenSearch data from backup: /opt/paragon/opensearch-backup/  
20250203_180002-priority  
2025-02-03 18:28:41 Restoring datastream template  
2025-02-03 18:28:42 Warning: Failed to create datastream template for jcloud_alerts_cleared.  
2025-02-03 18:28:42  
2025-02-03 18:28:42 Restoring datastream metadata  
2025-02-03 18:28:42 Warning: Failed to create datastream metadata for jcloud_alerts_cleared.  
2025-02-03 18:28:42  
2025-02-03 18:28:42 Normal indices Restore complete.  
2025-02-03 18:28:42 Restoring indices from datastream  
2025-02-03 18:28:42 Data Streams Restore complete.  
2025-02-03 18:28:42 Restoring ISM policy  
2025-02-03 18:28:43 Restoring ISM policy associations  
2025-02-03 18:28:44 Restore completed.
```

RELATED DOCUMENTATION

[OpenSearch Debugging Commands](#) | 37

CHAPTER 5

Troubleshoot Rook and Ceph

IN THIS CHAPTER

- [Rook-Ceph Failure | 45](#)

Rook-Ceph Failure

IN THIS SECTION

- [Problem | 45](#)
- [Cause | 45](#)
- [Solution | 46](#)

Problem

Sometimes when you are shutting down a node or a node fails, the pod status is stuck in initializing, Ceph status might hang, or the Rook module fails with the following error:

```
Error: "Module 'rook' has failed: None: Max retries exceeded with url: /api/v1/nodes (Caused by None)"
```

Cause

The Ceph monitor isn't fully initialized or ready to serve requests and the metadata server cannot complete its initialization because it's dependent on the Ceph monitor.

Solution

Restart rook-ceph-operator and the metadata server using the `kubectl rollout restart deployment -n rook-ceph rook-ceph-operator rook-ceph-mds-cephfs-a rook-ceph-mds-cephfs-b` command.

```
root@pal:~# kubectl rollout restart deployment -n rook-ceph rook-ceph-operator rook-ceph-mds-
cephfs-a rook-ceph-mds-cephfs-b
Warning: would violate PodSecurity "restricted:latest": allowPrivilegeEscalation != false
(container "rook-ceph-operator" must set securityContext.allowPrivilegeEscalation=false),
seccompProfile (pod or container "rook-ceph-operator" must set
securityContext.seccompProfile.type to "RuntimeDefault" or "Localhost")
deployment.apps/rook-ceph-operator restarted
Warning: would violate PodSecurity "restricted:latest": allowPrivilegeEscalation != false
(containers "chown-container-data-dir", "mds", "log-collector" must set
securityContext.allowPrivilegeEscalation=false), unrestricted capabilities (containers "chown-
container-data-dir", "mds", "log-collector" must set securityContext.capabilities.drop=["ALL"]),
restricted volume types (volumes "ceph-daemons-sock-dir", "rook-ceph-log", "rook-ceph-crash" use
restricted volume type "hostPath"), runAsNonRoot != true (pod or containers "chown-container-
data-dir", "mds", "log-collector" must set securityContext.runAsNonRoot=true), seccompProfile
(pod or containers "chown-container-data-dir", "mds", "log-collector" must set
securityContext.seccompProfile.type to "RuntimeDefault" or "Localhost")
deployment.apps/rook-ceph-mds-cephfs-a restarted
deployment.apps/rook-ceph-mds-cephfs-b restarted
```