Juniper Networks | Engineering Simplicity

Juniper Paragon Automation 2.3.0 Monitoring and Troubleshooting Guide

Published
2025-08-26

RELEASE
2.3.0

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 2.3.0 Monitoring and Troubleshooting Guide
2.3.0
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

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

CHAPTER 1

Monitor Using Sources and Sinks

IN THIS CHAPTER

Monitoring Overview

IN THIS SECTION

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

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

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

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

CHAPTER 2

Troubleshoot Using Paragon Shell

IN THIS CHAPTER

Troubleshooting Overview

IN THIS SECTION

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

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
`show paragon cluster nodes`	Shows node information of your Paragon Automation cluster.
`show paragon cluster pods`	Shows pod information of your Paragon Automation cluster.
`show paragon cluster namespaces`	Shows namespace information of your Paragon Automation cluster.
`show paragon cluster details`	Shows storage and controller node information of your Paragon Automation cluster.
`show paragon version`	Shows the version of your Paragon Automation cluster.
`show paragon images version`	Shows the version of pods in your Paragon Automation cluster.
`show paragon cluster pods namespace healthbot sort memory`	Shows the top pods of the `healthbot` namespace sorted by memory utilization.
`show paragon cluster pods namespace healthbot sort cpu`	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` *service-name*	Generates log files of different services within the `healthbot` namespace. Replace *service-name* with: • `tsdb-shim` • `tand` • `jtimon` • `config-server` • `api-server` • `analytical-engine` • `alerta`
`request paragon debug logs namespace foghorn service` *service-name*	Generates log files of different services within the `foghorn` namespace. Replace *service-name* with: • `order-management` • `placement` • `cmgd`
`request paragon debug logs namespace airflow service` *service-name*	Generates a log file of the `workflow-manager` service within the `airflow` namespace.
`request paragon debug logs namespace northstar service` *service-name*	Generates log files of different services within the `northstar` namespace. Replace *service-name* with: • `toposerver` • `web` • `configmonitor` • `api-aggregator`

Table 2: request paragon troubleshooting information Commands *(Continued)*

Command	Description
`request paragon debug logs namespace papi service` *`service-name`*	Generates log files of different services within the `papi` namespace. Replace *service-name* with: • `oc-term` • `papi` • `papi-ws`
`request paragon debug postgres`	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
Kafka	
`request paragon debug kafka ?`	Display possible completions for the `request paragon debug kafka` command.
`request paragon debug kafka options "-C -t` *`topic-name`* `-o s@`*`start-time`* `-o e@`*`end-time`* `-e -JB" output-file "`*`file-name`*`"`	Generate an output file of Kafka messages for a topic for a specified period of time.
Insights Kafka	

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 "device-id" time-period "duration"`	Display `insights-kafka-data` information for a device, for a specific time period. An output file of the information is generated.
Redis	
`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"`.
Service Logs	
`request paragon debug logs ?`	Display possible completions for the `request paragon debug logs` command.
`request paragon debug logs namespace name service service-name time duration`	Generate a log file for a service within a namespace for the specified time period.
TSDB	
`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
`request paragon debug get-tsdb-data device` *`device-id`* `topic "`*`topic-name`*`" output` *`file`*	Generate an output file of TSDB data for a particular device.
Postgres	
`request paragon debug postgres ?`	Display possible completions for the `request paragon debug postgres` command.
`request paragon debug postgres database` *`database-name`* `username` *`username`* `measurement` *`measurement-name`* `output (file)`	Generate an output file of the measurement value information of the Postgres database.

Perform a Health Check

IN THIS SECTION

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
```

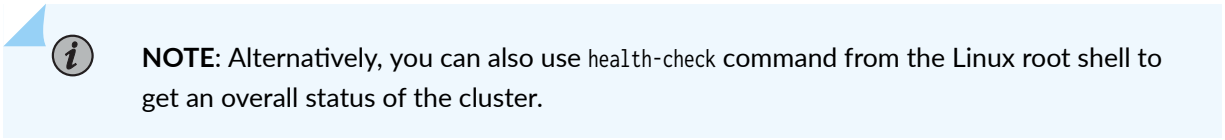
Meaning

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 that indicates that many health checks have failed and implies serious issues in the cluster. An amber status indicates that there maybe certain noncritical issues in the cluster. The status is returned 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



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

IN THIS SECTION

Purpose

Check utilization of local storage PVC, Ceph-based storage PVC, and the S3 bucket.

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
echolocal-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-9d9nd1l2                               /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 or paragon-ceph-usage-check --help

```
root@pa1:~# 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 or paragon-ceph-usage-check --s3

```
root@pa1:~# 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 or paragon-ceph-usage-check --cephfs

```
root@pa1:~# paragon-ceph-usage-check -c

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
|   0.0723346984013915  |
| 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 or paragon-ceph-usage-check --both

```
root@pa1:~# 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