Red Hat Satellite 6.8

Monitoring Red Hat Satellite

Collecting metrics from Red Hat Satellite 6

Last Updated: 2022-08-05

Red Hat Satellite 6.8 Monitoring Red Hat Satellite

Collecting metrics from Red Hat Satellite 6

Red Hat Satellite Documentation Team
satellite-doc-list@redhat.com

Legal Notice

Abstract

This guide describes how to gather metrics from Red Hat Satellite 6 for analysis. It is aimed at Satellite administrators.

Table of Contents

CHAPTER 1. OVERVIEW

Obtaining metrics from Satellite is useful for troubleshooting a current issue, and capacity planning. This guide describes how to collect live metrics and archive them for a fixed period of time. If you need to raise a support case with Red Hat to resolve a performance issue, the archived data provides valuable insight. Note that Red Hat Support can only access the archived data if you upload it to a Support Case.

You can collect the following metrics from Satellite:

- Basic statistics from Red Hat Enterprise Linux, including system load, memory utilization, and input/output operations;
- Process statistics, including memory and CPU utilization;
- Apache HTTP Server activity statistics;
- PostgreSQL activity statistics;
- Satellite application statistics.

Use Performance Co-Pilot (PCP) to collect and archive Satellite metrics.

CHAPTER 2. PERFORMANCE CO-PILOT

Performance Co-Pilot (PCP) is a suite of tools and libraries for acquiring, storing, and analyzing system-level performance measurements. PCP can be used to analyze live and historical metrics. Metrics can be retrieved and presented via the CLI, or a web UI.

2.1. PERFORMANCE METRIC DOMAIN AGENTS

A Performance Metric Domain Agent (PMDA) is a PCP add-on which enables access to metrics of an application or service. To gather all metrics relevant to Satellite, you must install PMDAs for Apache HTTP Server and PostgreSQL.

CHAPTER 3. INSTALLING PCP PACKAGES

This procedure describes how to install the PCP packages.

Prerequisites

- Ensure you have a minimum of 20 GB space available in the **/var/log/pcp** directory.
  The default PCP data retention policy is to retain only that data collected during the past 14 days. Data storage per day is estimated to use usually between 100 MB and 500 MB of disk space, but may use up to several gigabytes.
- Ensure that the base system on which Satellite Server is running is Red Hat Enterprise Linux 7.6. or later. The minimum supported version for the PCP packages is PCP version 4.1.

Procedure

1. Enable the Red Hat Enterprise Linux **optional** repository:

```
# subscription-manager repos --enable rhel-7-server-optional-rpms
```

2. Install the PCP packages:

```
# satellite-maintain packages install pcp \
  pcp-pmda-apache \
  pcp-pmda-postgresql \
  pcp-system-tools \
  pcp-webjs
```

3. Enable and start the Performance Metrics Collector daemon, and the Performance Metrics Logger daemon:

```
# systemctl enable pmcd pmlogger
# systemctl start pmcd pmlogger
```

3.1. CONFIGURING PCP DATA COLLECTION

This procedure describes how to configure PCP to collect metrics about processes, Satellite, Apache HTTP Server, and PostgreSQL.

Procedure

1. To configure PCP to collect data about Satellite processes, create the **/var/lib/pcp/pmdas/proc/hotproc.conf** file and include the following content:

```
#pmdahotproc
Version 1.0

# processes with load 0.9 or 1GB RSS memory
(cpuburn > 0.9 || residentsize > 1000000) &&
(
  fname == "java" ||
  fname == "qdrouterd" ||
  fname == "qpidd" ||
```

```
  (fname == "postgres" && psargs ~ /-D/) ||
  fname == "mongod" ||
  fname == "dynflow_executor" ||
  fname == "dynflow_executor_monitor" ||
  fname ~ /smart_proxy_dynflow_core/ ||
  psargs ~ /Passenger RackApp/ ||
  psargs ~ /celery beat/ ||
  psargs ~ /celery worker/ ||
  psargs ~ /pulp/ ||
  psargs ~ /smart-proxy/ ||
  psargs ~ /squid.conf/
)
```

By default, PCP collects basic system metrics. This step enables detailed metrics about the following Satellite processes:

- Java
- PostgreSQL
- MongoDB
- Dynflow
- Passenger
- Pulp
- Qpid

2. Configure PCP to log the process metrics being collected.

```
# mkdir -p /var/lib/pcp/config/pmlogconf/foreman-hotproc
# cat >/var/lib/pcp/config/pmlogconf/foreman-hotproc/summary << EOF
#pmlogconf-setup 2.0
ident   foreman hotproc metrics
probe   hotproc.control.config != "" ? include : exclude
        hotproc.psinfo.psargs
        hotproc.psinfo.cnswap
        hotproc.psinfo.nswap
        hotproc.psinfo.rss
        hotproc.psinfo.vsize
        hotproc.psinfo.cstime
        hotproc.psinfo.cutime
        hotproc.psinfo.stime
        hotproc.psinfo.utime
        hotproc.io.write_bytes
        hotproc.io.read_bytes
        hotproc.schedstat.cpu_time
        hotproc.fd.count
EOF
```

3. Install the process monitoring PMDA.

```
# cd /var/lib/pcp/pmdas/proc
# ./Install
```

4. Configure PCP to collect metrics from Apache HTTP Server.

   a. Enable the Apache HTTP Server extended status module.

```
#cat >/etc/httpd/conf.d/01-status.conf <<EOF
ExtendedStatus On
LoadModule status_module modules/mod_status.so

<Location "/server-status">
PassengerEnabled off
SetHandler server-status
Order deny,allow
Deny from all
Allow from localhost
</Location>
EOF
```

   b. Enable the Apache HTTP Server PMDA.

```
# cd /var/lib/pcp/pmdas/apache
# ./Install
```

   c. Prevent the Satellite installer overwriting the extended status module's configuration file. Add the following line to the **/etc/foreman-installer/custom-hiera.yaml** configuration file.

```
apache::purge_configs: false
```

5. Configure PCP to collect metrics from PostgreSQL.

   a. Change to the **/var/lib/pcp/pmdas/postgresql** directory.

```
# cd /var/lib/pcp/pmdas/postgresql
```

   b. Run the installer.

```
# ./Install
```

   c. Configure the PCP database interface to permit access to the PostgreSQL database. Edit the **/etc/pcpdbi.conf** configuration file, inserting the following lines:

```
$database = "dbi:Pg:dbname=foreman;host=localhost";
$username = "foreman";
$password = "6qXfN9m5nii5iEcbz8nuiJBNsyjjdRHA"; (1)
$os_user = "foreman";
```

   (1) The value for *$password* is stored in **/etc/foreman/database.yml** configuration file.

   d. Change the SELinux **pcp_pmcd_t** domain permission to permit PCP access to the PostgreSQL database.

```
# semanage permissive -a pcp_pmcd_t
```

e. Verify the PostgreSQL PMDA is able to connect to PostgreSQL.
Examine the **/var/log/pcp/pmcd/postgresql.log** file to confirm the connection is established. Without a successful database connection, the PostgreSQL PMDA will remain active, but not be able to provide any metrics.

```
[Tue Aug 14 09:21:06] pmdapostgresql(25056) Info: PostgreSQL connection established
```

If you find errors in **/var/log/pcp/pmcd/postgresql.log**, restart the **pmcd** service.

```
# systemctl restart pmcd
```

6. Enable telemetry functionality in Satellite.
To enable collection of metrics from Satellite, you must send metrics via the **statsd** protocol into the **pcp-mmvstatsd** daemon. The metrics are aggregated and available via the PCP MMV API.

a. Install the Foreman Telemetry and **pcp-mmvstatsd** packages.

```
# satellite-maintain packages install foreman-telemetry pcp-mmvstatsd
```

b. Enable and start the **pcp-mmvstatsd** service.

```
# systemctl enable pcp-mmvstatsd
# systemctl start pcp-mmvstatsd
```

c. Enable the Satellite telemetry functionality.
Add the following lines to **/etc/foreman/settings.yaml** configuration file:

```
:telemetry:
  :prefix: 'fm_rails'
  :statsd:
    :enabled: true
    :host: '127.0.0.1:8125'
    :protocol: 'statsd'
  :prometheus:
    :enabled: false
  :logger:
    :enabled: false
    :level: 'INFO'
```

7. Schedule daily storage of metrics in archive files:

```
# cat >/etc/cron.daily/refresh_mmv <<EOF
#!/bin/bash
echo "log mandatory on 1 minute mmv" | /usr/bin/pmlc -P
EOF
# chmod +x /etc/cron.daily/refresh_mmv
```

8. Restart the Apache HTTP Server and PCP to begin data collection:

```
# systemctl restart httpd pmcd pmlogger
```

3.2. ENABLING ACCESS TO METRICS VIA THE WEB UI

This procedure describes how to access metrics collected by PCP, via the web UI.

Procedure

1. Enable the Red Hat Enterprise Linux **optional** repository:

```
# subscription-manager repos --enable rhel-7-server-optional-rpms
```

2. Install the PCP web API and applications:

```
# satellite-maintain packages install pcp-webapi pcp-webapp-grafana pcp-webapp-vector
```

3. Start and enable the PCP web service:

```
# systemctl start pmwebd
# systemctl enable pmwebd
```

4. Open firewall port to allow access to the PCP web service:

```
# firewall-cmd --add-port=44323/tcp
# firewall-cmd --permanent --add-port=44323/tcp
```

3.3. VERIFYING PCP CONFIGURATION

To verify PCP is configured correctly, and services are active, run the following command:

```
# pcp
```

This outputs a summary of the active PCP configuration.

Example output from the pcp command:

```
Performance Co-Pilot configuration on satellite.example.com:

 platform: Linux satellite.example.com 3.10.0-862.3.3.el7.x86_64 #1 SMP Wed Jun 13 05:44:23 EDT
2018 x86_64
 hardware: 8 cpus, 4 disks, 1 node, 23380MB RAM
 timezone: AEST-10
 services: pmcd pmwebd
     pmcd: Version 3.12.2-1, 9 agents, 1 client
     pmda: root pmcd proc xfs linux apache mmv postgresql jbd2
 pmlogger: primary logger: /var/log/pcp/pmlogger/satellite.example.com/20180802.00.10
```

In this example, both the Performance Metrics Collector Daemon (pmcd), and the Performance Metrics Web Daemon (pmwebd) services are running. It also confirms the PMDAs which are collecting metrics. Finally, it lists the currently actively archive file, in which **pmlogger** is storing metrics.

CHAPTER 4. PCP METRICS

Metrics are stored in a tree-like structure. For example, all network metrics are stored in a node named **network**. Each metric may be a single value, or a list of values, known as instances. For example, kernel load has three instances, a 1-minute, 5-minute, and 15-minute average.

For every metric entry, PCP stores both its data and metadata. This includes the metrics description, data type, units, and dimensions. For example, the metadata enables PCP to output multiple metrics with different dimensions.

The value of a counter metric only increases. For example, a count of disk write operations on a specific device only increases. When you query the value of a counter metric, PCP converts this into a rate value by default.

In addition to system metrics such as CPU, memory, kernel, XFS, disk, and network, the following metrics are configured:

Metric	Description
hotproc.*	Basic metrics of key Satellite processes
apache.*	Apache HTTP Server metrics
postgresql.*	Basic PostgreSQL statistics
mmv.fm_rails_*	Satellite metrics

4.1. IDENTIFYING AVAILABLE METRICS

- To list all metrics available via PCP, enter the following command:

```
# pminfo
```

- To list all Satellite metrics and their descriptions, enter the following command:

```
# foreman-rake telemetry:metrics
```

- To list the archived metrics, enter the following command:

```
# less /var/log/pcp/pmlogger/$(hostname)/pmlogger.log
```

- The pmlogger daemon archives data as it is received, according to its configuration. To confirm the active archive file, enter the following command:

```
# pcp | grep logger
```

  The output includes the file name of the active archive file, for example:

```
/var/log/pcp/pmlogger/satellite.example.com/20180814.00.10
```

CHAPTER 5. RETRIEVING METRICS

You can retrieve metrics from PCP using the CLI or the web UI interfaces. A number of CLI tools are provided with PCP, which can either output live data, or data from archived sources. The web UI interfaces are provided by the Grafana and Vector web applications. Vector connects directly to the PCP daemon, and can only display live data. Grafana reads from PCP archive files and can display data to up to 1 year old.

5.1. RETRIEVING METRICS VIA THE CLI

Using the CLI tools provided with PCP, you can retrieve metrics either live, or from an archive file.

5.1.1. Retrieving Live Metrics using CLI

To output metrics on disk partition write instances, enter the following command:

```
# pmval -f 1 disk.partitions.write
```

In this example, PCP converts the number of writes to disk partitions from a counter value, to a rate value. The **-f 1** specifies that the value be abbreviated to one decimal place.

Example output

```
metric:    disk.partitions.write
host:      satellite.example.com
semantics: cumulative counter (converting to rate)
units:     count (converting to count / sec)
samples:   all

              vda1             vda2              sr0
               0.0             12.0              0.0
               0.0              1.0              0.0
               0.0              1.0              0.0
               0.0              2.0              0.0
```

To monitor system metrics with a two second interval:

```
# pmstat -t 2sec
```

5.1.2. Retrieving Archived Metrics using CLI

You can use the PCP CLI tools to retrieve metrics from an archive file. To do that, add the **--archive** parameter and specify the archive file.

- To list all metrics which were enabled when the archive file was created, enter the following command:

  ```
  pminfo --archive archive_file
  ```

- To confirm the host and time period covered by an archive file, enter the following command:

  ```
  # pmdumplog -l archive_file
  ```

Examples

- To list disk writes for each partition, over the time period covered by the archive file:

```
# pmval --archive /var/log/pcp/pmlogger/satellite.example.com/20180816.00.10 \
  -f 1 disk.partitions.write
```

- To list disk write operations per partition, with a two second interval, between the time period 14:00 and 14:15:

```
# pmval --archive /var/log/pcp/pmlogger/satellite.example.com/20180816.00.10 \
  -d -t 2sec \
  -f 3 disk.partitions.write \
  -S @14:00 -T @14:15
```

- To list average values of all performance metrics, including the time of minimum/maximum value and the actual minimum/maximum value, between the time period 14:00 and 14:30. To output the values in tabular formatting:

```
# pmlogsummary /var/log/pcp/pmlogger/satellite.example.com/20180816.00.10 \
  -HlfiImM \
  -S @14:00 \
  -T @14:30 \
  disk.partitions.write \
  mem.freemem
```

- To list system metrics stored in an archive, starting from 14:00. The metrics are displayed in a format similar to the **top** tool.

```
# pcp --archive /var/log/pcp/pmlogger/satellite.example.com/20180816.00.10 \
  -S @14:00 \
  atop
```

5.2. RETRIEVING METRICS IN THE WEB UI

The Vector and Grafana web applications provide a dashboard-style view, with default widgets displaying the values of metrics. You can add and remove metrics to suit your requirements.

You can also select the time span shown for each widget. Only Grafana provides the option of selecting a custom time range from the archived metrics.

For more details on using Grafana, see the Grafana Labs website. For more details on using Vector, see the Vector website.

Procedure

To retrieve metrics in the Performance Co-Pilot web applications UI, complete the following steps:

1. Open the following URL: http://*satellite.example.com*:44323
2. In the PCP web UI, select the web application dashboard that you want to access.

Figure 5.1. Example of the Grafana dashboard

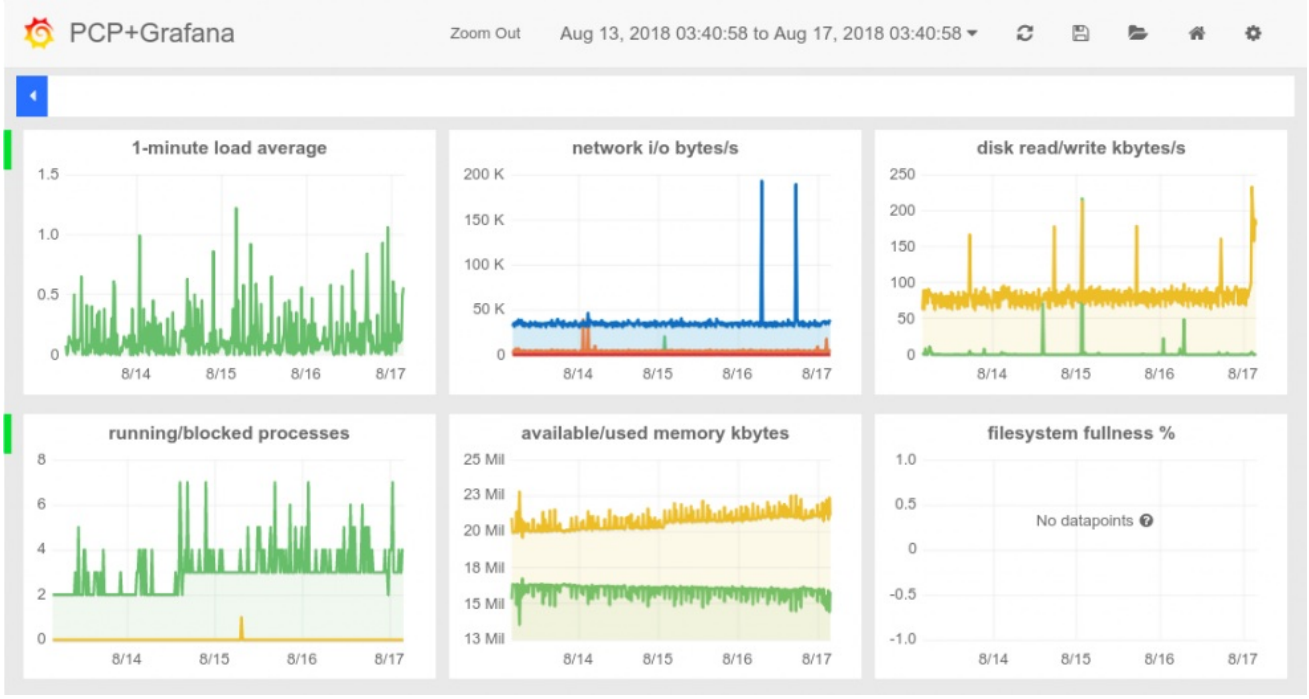
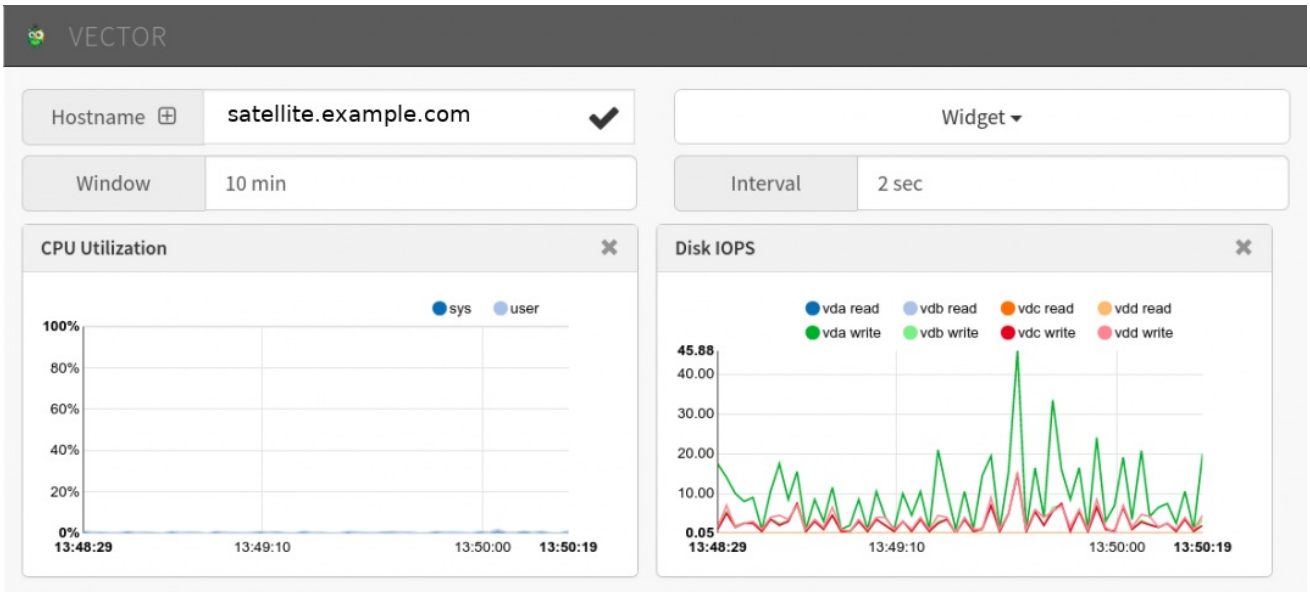


Figure 5.2. Example of the Vector dashboard



CHAPTER 6. METRICS DATA RETENTION

The storage capacity required by PCP data logging is determined by the following factors:

- the metrics being logged,
- the logging interval,
- and the retention policy.

The default logging (sampling) interval is 60 seconds.

The default retention policy is to keep archives for the last 14 days, compressing archives older than one day. PCP archive logs are stored in the **/var/log/pcp/pmlogger/*hostname*** directory.

6.1. CHANGING DEFAULT LOGGING INTERVAL

This procedure describes how to change the default logging interval.

Procedure

1. Edit the **/etc/pcp/pmlogger/control.d/local** configuration file.
2. Edit the LOCALHOSTNAME line and append **-t *XX*s**, where *XX* is the desired time interval, measured in seconds.
3. Restart the **pmlogger** service.

6.2. CHANGING DATA RETENTION POLICY

This procedure describes how to change the data retention policy.

Procedure

1. Edit the **/etc/cron.d/pcp-pmlogger** file.
2. Find the line containing **pmlogger_daily**.
3. Change the value for parameter **-x** to the desired number of days after which data is archived.
4. Add parameter **-k**, and add a value for the number of days after which data is deleted.
   For example, the parameters **-x 4 -k 7** specify that data will be compressed after 4 days, and deleted after 7 days.

6.3. CONFIRMING DATA STORAGE USAGE

To confirm data storage usage, enter the following command:

```
# less /var/log/pcp/pmlogger/$(hostname)/pmlogger.log
```

This lists all available metrics, grouped by the frequency at which they are logged. For each group it also lists the storage required to store the listed metrics, per day.

Example storage statistics

logged every 60 sec: 61752 bytes or 84.80 Mbytes/day