Red Hat Enterprise Linux 9.0 Beta

Monitoring and managing system status and performance

Optimizing system throughput, latency, and power consumption
Red Hat Enterprise Linux 9.0 Beta Monitoring and managing system status and performance

Optimizing system throughput, latency, and power consumption
Abstract

This documentation collection provides instructions on how to monitor and optimize the throughput, latency, and power consumption of Red Hat Enterprise Linux 9 in different scenarios.
Table of Contents

RHEL BETA RELEASE ................................................................. 4
MAKING OPEN SOURCE MORE INCLUSIVE ...................................... 5
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION ....................... 6

CHAPTER 1. MONITORING PERFORMANCE USING RHEL SYSTEM ROLES ............ 7
1.1. INTRODUCTION TO RHEL SYSTEM ROLES .................................. 7
1.2. RHEL SYSTEM ROLES TERMINOLOGY ....................................... 7
1.3. INSTALLING RHEL SYSTEM ROLES IN YOUR SYSTEM ...................... 8
1.4. APPLYING A ROLE ............................................................. 8
1.5. INTRODUCTION TO THE METRICS SYSTEM ROLE ......................... 11
1.6. USING THE METRICS SYSTEM ROLE TO MONITOR YOUR LOCAL SYSTEM WITH VISUALIZATION 12
1.7. USING THE METRICS SYSTEM ROLE TO SETUP A FLEET OF INDIVIDUAL SYSTEMS TO MONITOR THEMSELVES 12
1.8. USING THE METRICS SYSTEM ROLE TO MONITOR A FLEET OF MACHINES CENTRALLY VIA YOUR LOCAL MACHINE 13
1.9. SETTING UP AUTHENTICATION WHILE MONITORING A SYSTEM USING THE METRICS SYSTEM ROLE 14
1.10. USING THE METRICS SYSTEM ROLE TO CONFIGURE AND ENABLE METRICS COLLECTION FOR SQL SERVER 14

CHAPTER 2. SETTING UP PCP .......................................................... 17
2.1. OVERVIEW OF PCP ............................................................ 17
2.2. INSTALLING AND ENABLING PCP ............................................. 17
2.3. DEPLOYING A MINIMAL PCP SETUP ........................................ 18
2.4. SYSTEM SERVICES DISTRIBUTED WITH PCP ............................... 19
2.5. TOOLS DISTRIBUTED WITH PCP ............................................ 20
2.6. PCP DEPLOYMENT ARCHITECTURES ....................................... 23
2.7. RECOMMENDED DEPLOYMENT ARCHITECTURE .......................... 26
2.8. SIZING FACTORS .............................................................. 26
2.9. CONFIGURATION OPTIONS FOR PCP SCALING ............................ 27
2.10. EXAMPLE: ANALYZING THE CENTRALIZED LOGGING DEPLOYMENT ... 27
2.11. EXAMPLE: ANALYZING THE FEDERATED SETUP DEPLOYMENT ........ 28
2.12. TROUBLESHOOTING HIGH MEMORY USAGE ................................. 29

CHAPTER 3. LOGGING PERFORMANCE DATA WITH PMLOGGER ....................... 32
3.1. MODIFYING THE PMLOGGER CONFIGURATION FILE WITH PMLOGCONF ... 32
3.2. EDITING THE PMLOGGER CONFIGURATION FILE MANUALLY ............. 32
3.3. ENABLING THE PMLOGGER SERVICE ....................................... 33
3.4. SETTING UP A CLIENT SYSTEM FOR METRICS COLLECTION ............ 34
3.5. SETTING UP A CENTRAL SERVER TO COLLECT DATA .................... 35
3.6. REPLAYING THE PCP LOG ARCHIVES WITH PMDUMPTEXT ................ 36

CHAPTER 4. MONITORING PERFORMANCE WITH PERFORMANCE CO-PILOT ............ 38
4.1. MONITORING POSTFIX WITH PMDA-POSTFIX ................................ 38
4.2. VISUALLY TRACING PCP LOG ARCHIVES WITH THE PCP CHARTS APPLICATION 39
4.3. COLLECTING DATA FROM SQL SERVER USING PCP ......................... 41

CHAPTER 5. PERFORMANCE ANALYSIS OF XFS WITH PCP .......................... 44
5.1. INSTALLING XFS PMDA MANUALLY .......................................... 44
5.2. EXAMINING XFS PERFORMANCE METRICS WITH PMINFO ............ 44
5.3. RESETTING XFS PERFORMANCE METRICS WITH PMSTORE ............. 46
5.4. PCP METRIC GROUPS FOR XFS ............................................. 46
CHAPTER 6. SETTING UP GRAPHICAL REPRESENTATION OF PCP METRICS

6.1. SETTING UP PCP WITH PCP-ZEROCONF
6.2. SETTING UP A GRAFANA-SERVER
6.3. ACCESSING THE GRAFANA WEB UI
6.4. CONFIGURING PCP REDIS
6.5. CREATING PANELS AND ALERT IN PCP REDIS DATA SOURCE
6.6. ADDING NOTIFICATION CHANNELS FOR ALERTS
6.7. SETTING UP AUTHENTICATION BETWEEN PCP COMPONENTS
6.8. INSTALLING PCP BPFTRACE
6.9. VIEWING THE PCP BPFTRACE SYSTEM ANALYSIS DASHBOARD
6.10. INSTALLING PCP VECTOR
6.11. VIEWING THE PCP VECTOR CHECKLIST
6.12. TROUBLESHOOTING GRAFANA ISSUES
RHEL BETA RELEASE

Red Hat provides Red Hat Enterprise Linux Beta access to all subscribed Red Hat accounts. The purpose of Beta access is to:

- Provide an opportunity to customers to test major features and capabilities prior to the general availability release and provide feedback or report issues.

- Provide Beta product documentation as a preview. Beta product documentation is under development and is subject to substantial change.

Note that Red Hat does not support the usage of RHEL Beta releases in production use cases. For more information, see What does Beta mean in Red Hat Enterprise Linux and can I upgrade a RHEL Beta installation to a General Availability (GA) release?
MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better. To do so:

- For simple comments on specific passages:
  1. Make sure you are viewing the documentation in the Multi-page HTML format. In addition, ensure you see the Feedback button in the upper right corner of the document.
  2. Use your mouse cursor to highlight the part of text that you want to comment on.
  3. Click the Add Feedback pop-up that appears below the highlighted text.
  4. Follow the displayed instructions.

- For submitting more complex feedback, create a Bugzilla ticket:
  1. Go to the Bugzilla website.
  2. As the Component, use Documentation.
  3. Fill in the Description field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. Click Submit Bug.
CHAPTER 1. MONITORING PERFORMANCE USING RHEL SYSTEM ROLES

As a system administrator, you can use the metrics RHEL System Role to monitor the performance of a system.

1.1. INTRODUCTION TO RHEL SYSTEM ROLES

RHEL System Roles is a collection of Ansible roles and modules. RHEL System Roles provide a configuration interface to remotely manage multiple RHEL systems. The interface enables managing system configurations across multiple versions of RHEL, as well as adopting new major releases.

On Red Hat Enterprise Linux 9, the interface currently consists of the following roles:

- network
- certificate
- postfix
- kernel_settings
- metrics
- nbde_client and nbde_server
- tlog
- ssh
- sshd
- crypto_policies

All these roles are provided by the rhel-system-roles package available in the AppStream repository.

Additional resources

- Red Hat Enterprise Linux (RHEL) System Roles
- /usr/share/doc/rhel-system-roles documentation [1]

1.2. RHEL SYSTEM ROLES TERMINOLOGY

You can find the following terms across this documentation:

System Roles terminology

Ansible playbook

Playbooks are Ansible's configuration, deployment, and orchestration language. They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process.

Control node

Any machine with Ansible installed. You can run commands and playbooks, invoking /usr/bin/ansible
or /usr/bin/ansible-playbook, from any control node. You can use any computer that has Python installed on it as a control node - laptops, shared desktops, and servers can all run Ansible. However, you cannot use a Windows machine as a control node. You can have multiple control nodes.

Inventory
A list of managed nodes. An inventory file is also sometimes called a “hostfile”. Your inventory can specify information like IP address for each managed node. An inventory can also organize managed nodes, creating and nesting groups for easier scaling. To learn more about inventory, see the Working with Inventory section.

Managed nodes
The network devices, servers, or both that you manage with Ansible. Managed nodes are also sometimes called “hosts”. Ansible is not installed on managed nodes.

1.3. INSTALLING RHEL SYSTEM ROLES IN YOUR SYSTEM

To use the RHEL System Roles, install the required packages in your system.

Prerequisites
- You have Ansible packages installed in the system you want to use as a control node:

Procedure
1. Install the `rhel-system-roles` package on the system that you want to use as a control node:
   
   ```
   # yum install rhel-system-roles
   ```

2. Install the `ansible-core` package:

   ```
   # yum install ansible-core
   ```

   The `ansible-core` package provides the `ansible-playbook` CLI, the Ansible Vault functionality, and the basic modules and filters required by RHEL Ansible content.

As a result, you are able to create an Ansible playbook.

Additional resources
- The Red Hat Enterprise Linux (RHEL) System Roles
- The `ansible-playbook` man page.

1.4. APPLYING A ROLE

The following procedure describes how to apply a particular role.

Prerequisites
- Ensure that the `rhel-system-roles` package is installed on the system that you want to use as a control node:

  ```
  # yum install rhel-system-roles
  ```
1. Install the **ansible-core** package:

```
# yum install ansible-core
```

The **ansible-core** package provides the **ansible-playbook** CLI, the Ansible Vault functionality, and the basic modules and filters required by RHEL Ansible content.

- Ensure that you are able to create an Ansible inventory.
  Inventories represent the hosts, host groups, and some of the configuration parameters used by the Ansible playbooks.

  Playbooks are typically human-readable, and are defined in **ini**, **yaml**, **json**, and other file formats.

- Ensure that you are able to create an Ansible playbook.
  Playbooks represent Ansible’s configuration, deployment, and orchestration language. By using playbooks, you can declare and manage configurations of remote machines, deploy multiple remote machines or orchestrate steps of any manual ordered process.

  A playbook is a list of one or more **plays**. Every **play** can include Ansible variables, tasks, or roles.

  Playbooks are human-readable, and are defined in the **yaml** format.

**Procedure**

1. Create the required Ansible inventory containing the hosts and groups that you want to manage. Here is an example using a file called **inventory.ini** of a group of hosts called **webservers**:

   ```ini
   [webservers]
   host1
   host2
   host3
   ```

2. Create an Ansible playbook including the required role. The following example shows how to use roles through the **roles**: option for a playbook:

   The following example shows how to use roles through the **roles**: option for a given **play**:

   ```yaml
   - hosts: webservers
     roles:
       - rhel-system-roles.network
       - rhel-system-roles.postfix
   ```
NOTE

Every role includes a README file, which documents how to use the role and supported parameter values. You can also find an example playbook for a particular role under the documentation directory of the role. Such documentation directory is provided by default with the `rhel-system-roles` package, and can be found in the following location:

`/usr/share/doc/rhel-system-roles/SUBSYSTEM/`

Replace `SUBSYSTEM` with the name of the required role, such as `postfix`, `metrics`, `network`, `tlog`, or `ssh`.

3. To execute the playbook on specific hosts, you must perform one of the following:

   - Edit the playbook to use `hosts: host1[,host2,...]`, or `hosts: all`, and execute the command:
     ```bash
     # ansible-playbook name.of.the.playbook
     ```
   
   - Edit the inventory to ensure that the hosts you want to use are defined in a group, and execute the command:
     ```bash
     # ansible-playbook -i name.of.the.inventory name.of.the.playbook
     ```
   
   - Specify all hosts when executing the `ansible-playbook` command:
     ```bash
     # ansible-playbook -i host1,host2,... name.of.the.playbook
     ```

IMPORTANT

Be aware that the `-i` flag specifies the inventory of all hosts that are available. If you have multiple targeted hosts, but want to select a host against which you want to run the playbook, you can add a variable in the playbook to be able to select a host. For example:

Ansible Playbook | example-playbook.yml:

```yaml
- hosts: "{{ target_host }}"
  roles:
    - rhel-system-roles.network
    - rhel-system-roles.postfix
```

Playbook execution command:

```bash
# ansible-playbook -i host1,,hostn -e target_host=host5 example-playbook.yml
```

Additional resources

- Ansible playbooks
- Using roles in Ansible playbook
1.5. INTRODUCTION TO THE METRICS SYSTEM ROLE

RHEL System Roles is a collection of Ansible roles and modules that provide a consistent configuration interface to remotely manage multiple RHEL systems. The metrics System Role configures performance analysis services for the local system and, optionally, includes a list of remote systems to be monitored by the local system. The metrics System Role enables you to use `pcp` to monitor your systems performance without having to configure `pcp` separately, as the set-up and deployment of `pcp` is handled by the playbook.

Table 1.1. Metrics system role variables

<table>
<thead>
<tr>
<th>Role variable</th>
<th>Description</th>
<th>Example usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>metrics_monitored_hosts</td>
<td>List of remote hosts to be analyzed by the target host. These hosts will have metrics recorded on the target host, so ensure enough disk space exists below <code>/var/log</code> for each host.</td>
<td><code>metrics_monitored_hosts: [&quot;webserver.example.com&quot;, &quot;database.example.com&quot;]</code></td>
</tr>
<tr>
<td>metrics_retention_days</td>
<td>Configures the number of days for performance data retention before deletion.</td>
<td><code>metrics_retention_days: 14</code></td>
</tr>
<tr>
<td>metrics_graph_service</td>
<td>A boolean flag that enables the host to be set up with services for performance data visualization via <code>pcp</code> and <code>grafana</code>. Set to false by default.</td>
<td><code>metrics_graph_service: false</code></td>
</tr>
<tr>
<td>metrics_query_service</td>
<td>A boolean flag that enables the host to be set up with time series query services for querying recorded <code>pcp</code> metrics via <code>redis</code>. Set to false by default.</td>
<td><code>metrics_query_service: false</code></td>
</tr>
<tr>
<td>metrics_provider</td>
<td>Specifies which metrics collector to use to provide metrics. Currently, <code>pcp</code> is the only supported metrics provider.</td>
<td><code>metrics_provider: &quot;pcp&quot;</code></td>
</tr>
</tbody>
</table>

NOTE

For details about the parameters used in `metrics_connections` and additional information about the metrics System Role, see the `/usr/share/ansible/roles/rhel-system-roles.metrics/README.md` file.
1.6. USING THE METRICS SYSTEM ROLE TO MONITOR YOUR LOCAL SYSTEM WITH VISUALIZATION

This procedure describes how to use the metrics RHEL System Role to monitor your local system while simultaneously provisioning data visualization via grafana.

Prerequisites

- The ansible-core package is installed on the control machine.
- You have the rhel-system-roles package installed on the machine you want to monitor.

Procedure

1. Configure localhost in the /etc/ansible/hosts Ansible inventory by adding the following content to the inventory:

   localhost ansible_connection=local

2. Create an Ansible playbook with the following content:

   ```yaml
   ---
   - hosts: localhost
     vars:
       metrics_graph_service: yes
     roles:
     - rhel-system-roles.metrics
   ```

3. Run the Ansible playbook:

   ```bash
   # ansible-playbook name_of_your_playbook.yml
   ```

   **NOTE**

   Since the metrics_graph_service boolean is set to value="yes", grafana is automatically installed and provisioned with pcp added as a data source.

4. To view visualization of the metrics being collected on your machine, access the grafana web interface as described in Accessing the Grafana web UI.

1.7. USING THE METRICS SYSTEM ROLE TO SETUP A FLEET OF INDIVIDUAL SYSTEMS TO MONITOR THEMSELVES

This procedure describes how to use the metrics System Role to set up a fleet of machines to monitor themselves.

Prerequisites

- The ansible-core package is installed on the control machine.
- You have the rhel-system-roles package installed on the machine you want to use to run the playbook.
Procedure

1. Add the name or IP of the machines you wish to monitor via the playbook to the `/etc/ansible/hosts` Ansible inventory file under an identifying group name enclosed in brackets:

   ```
   [remotes]
   webserver.example.com
database.example.com
   ```

2. Create an Ansible playbook with the following content:

   ```
   ---
   - hosts: remotes
     vars:
       metrics_retention_days: 0
     roles:
       - rhel-system-roles.metrics
   ```

3. Run the Ansible playbook:

   ```
   # ansible-playbook name_of_your_playbook.yml
   ```

1.8. USING THE METRICS SYSTEM ROLE TO MONITOR A FLEET OF MACHINES CENTRALLY VIA YOUR LOCAL MACHINE

This procedure describes how to use the metrics System Role to set up your local machine to centrally monitor a fleet of machines while also provisioning visualization of the data via grafana and querying of the data via redis.

Prerequisites

- The `ansible-core` package is installed on the control machine.
- You have the `rhel-system-roles` package installed on the machine you want to use to run the playbook.

Procedure

1. Create an Ansible playbook with the following content:

   ```
   ---
   - hosts: localhost
     vars:
       metrics_graph_service: yes
       metrics_query_service: yes
       metrics_retention_days: 10
       metrics_monitored_hosts: ["database.example.com", "webserver.example.com"]
     roles:
       - rhel-system-roles.metrics
   ```

2. Run the Ansible playbook:

   ```
   # ansible-playbook name_of_your_playbook.yml
   ```
NOTE

Since the `metrics_graph_service` and `metrics_query_service` booleans are set to value="yes", grafana is automatically installed and provisioned with `pcp` added as a data source with the `pcp` data recording indexed into `redis`, allowing the `pcp` querying language to be used for complex querying of the data.

3. To view graphical representation of the metrics being collected centrally by your machine and to query the data, access the grafana web interface as described in Accessing the Grafana web UI.

1.9. SETTING UP AUTHENTICATION WHILE MONITORING A SYSTEM USING THE METRICS SYSTEM ROLE

PCP supports the `scram-sha-256` authentication mechanism through the Simple Authentication Security Layer (SASL) framework. The metrics RHEL System Role automates the steps to setup authentication using the `scram-sha-256` authentication mechanism. This procedure describes how to setup authentication using the metrics RHEL System Role.

Prerequisites

- The `ansible-core` package is installed on the control machine.
- You have the `rhel-system-roles` package installed on the machine you want to use to run the playbook.

Procedure

1. Include the following variables in the Ansible playbook you want to setup authentication for:

   ```yaml
   ---
   vars:
   metrics_username: your_username
   metrics_password: your_password
   ```

2. Run the Ansible playbook:

   ```bash
   # ansible-playbook name_of_your_playbook.yml
   ```

Verification steps

- Verify the `sasl` configuration:

  ```bash
  # pminfo -f -h "pcp://127.0.0.1?username=your_username" disk.dev.read
  Password:
  disk.dev.read
  inst [0 or "sda"] value 19540
  ```

1.10. USING THE METRICS SYSTEM ROLE TO CONFIGURE AND ENABLE METRICS COLLECTION FOR SQL SERVER

This procedure describes how to use the metrics RHEL System Role to automate the configuration and enabling of metrics collection for Microsoft SQL Server via `pcp` on your local system.
Prerequisites

- The **ansible-core** package is installed on the control machine.
- You have the **rhel-system-roles** package installed on the machine you want to monitor.
- You have installed Microsoft SQL Server for Red Hat Enterprise Linux and established a ‘trusted’ connection to an SQL server.
- You have installed the Microsoft ODBC driver for SQL Server for Red Hat Enterprise Linux.

Procedure

1. Configure **localhost** in the the `/etc/ansible/hosts` Ansible inventory by adding the following content to the inventory:

   ```bash
   localhost ansible_connection=local
   ```

2. Create an Ansible playbook that contains the following content:

   ```yaml
   ---
   - hosts: localhost
     roles:
       - role: rhel-system-roles.metrics
     vars:
       metrics_from_mssql: yes
   ```

3. Run the Ansible playbook:

   ```bash
   # ansible-playbook name_of_your_playbook.yml
   ```

Verification steps

- Use the **pcp** command to verify that SQL Server PMDA agent (mssql) is loaded and running:

  ```bash
  # pcp
  platform: Linux rhel82-2.local 4.18.0-167.el8.x86_64 #1 SMP Sun Dec 15 01:24:23 UTC 2019 x86_64
  hardware: 2 cpus, 1 disk, 1 node, 2770MB RAM
  timezone: PDT+7
  services: pmcd pmproxy
  pmda: root pmcd proc pmproxy xfs linux nfsclient mmv kvm mssql jbd2 dm
  pmlogger: primary logger: /var/log/pcp/pmlogger/rhel82-2.local/20200326.16.31
  pmie: primary engine: /var/log/pcp/pmie/rhel82-2.local/pmie.log
  ```

Additional resources

- For more information about using Performance Co-Pilot for Microsoft SQL Server, see this Red Hat Developers Blog post.
This documentation is installed automatically with the `rhel-system-roles` package.
CHAPTER 2. SETTING UP PCP

Performance Co-Pilot (PCP) is a suite of tools, services, and libraries for monitoring, visualizing, storing, and analyzing system-level performance measurements.

This section describes how to install and enable PCP on your system.

2.1. OVERVIEW OF PCP

You can add performance metrics using Python, Perl, C++, and C interfaces. Analysis tools can use the Python, C++, C client APIs directly, and rich web applications can explore all available performance data using a JSON interface.

You can analyze data patterns by comparing live results with archived data.

Features of PCP:

- Light-weight distributed architecture, which is useful during the centralized analysis of complex systems.
- It allows the monitoring and management of real-time data.
- It allows logging and retrieval of historical data.

PCP has the following components:

- The Performance Metric Collector Daemon (pmcd) collects performance data from the installed Performance Metric Domain Agents (pmda). PMDAs can be individually loaded or unloaded on the system and are controlled by the PMCD on the same host.
- Various client tools, such as pminfo or pmstat, can retrieve, display, archive, and process this data on the same host or over the network.
- The pcp package provides the command-line tools and underlying functionality.
- The pcp-gui package provides the graphical application. Install the pcp-gui package by executing the yum install pcp-gui command. For more information, see Visually tracing PCP log archives with the PCP Charts application.

Additional resources

- pcp(1) man page
- /usr/share/doc/pcp-doc/ directory
- Tools distributed with PCP
- Index of Performance Co-Pilot (PCP) articles, solutions, tutorials, and white papers fromon Red Hat Customer Portal
- Side-by-side comparison of PCP tools with legacy tools Red Hat Knowledgebase article
- PCP upstream documentation

2.2. INSTALLING AND ENABLING PCP
To begin using PCP, install all the required packages and enable the PCP monitoring services.

This procedure describes how to install PCP using the `pcp` package. If you want to automate the PCP installation, install it using the `pcp-zeroconf` package. For more information on installing PCP by using `pcp-zeroconf`, see Setting up PCP with pcp-zeroconf.

**Procedure**

1. Install the `pcp` package:
   ```
   # yum install pcp
   ```

2. Enable and start the `pmcd` service on the host machine:
   ```
   # systemctl enable pmcd
   # systemctl start pmcd
   ```

**Verification steps**

- Verify if the `pmcd` process is running on the host:
  ```
  # pcp
  ```

Performance Co-Pilot configuration on workstation:

- platform: Linux workstation 4.18.0-80.el8.x86_64 #1 SMP Wed Mar 13 12:02:46 UTC 2019 x86_64
- hardware: 12 cpus, 2 disks, 1 node, 36023MB RAM
- timezone: CEST-2
- services: pmcd
- pmcd: Version 4.3.0-1, 8 agents
- pmda: root pmcd proc xfs linux mmv kvm jbd2

**Additional resources**

- `pmcd(1)` man page
- Tools distributed with PCP

### 2.3. DEPLOYING A MINIMAL PCP SETUP

The minimal PCP setup collects performance statistics on Red Hat Enterprise Linux. The setup involves adding the minimum number of packages on a production system needed to gather data for further analysis.

You can analyze the resulting `tar.gz` file and the archive of the `pmlogger` output using various PCP tools and compare them with other sources of performance information.

**Prerequisites**

- PCP is installed. For more information, see Installing and enabling PCP.
Procedure

1. Update the `pmlogger` configuration:
   
   ```bash
   # pmlogconf -r /var/lib/pcp/config/pmlogger/config.default
   ```

2. Start the `pmcd` and `pmlogger` services:
   
   ```bash
   # systemctl start pmcd.service
   # systemctl start pmlogger.service
   ```

3. Execute the required operations to record the performance data.

4. Stop the `pmcd` and `pmlogger` services:
   
   ```bash
   # systemctl stop pmcd.service
   # systemctl stop pmlogger.service
   ```

5. Save the output and save it to a `.tar.gz` file named based on the host name and the current date and time:
   
   ```bash
   # cd /var/log/pcp/pmlogger/
   # tar -czf $(hostname).$(date +%F-%Hh%M).pcp.tar.gz $(hostname)
   ```
   
   Extract this file and analyze the data using PCP tools.

Additional resources

- `pmlogconf(1)`, `pmlogger(1)`, and `pmcd(1)` man pages
- Tools distributed with PCP
- System services distributed with PCP

2.4. SYSTEM SERVICES DISTRIBUTED WITH PCP

The following table describes roles of various system services, which are distributed with PCP.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pmcd</td>
<td>The Performance Metric Collector Daemon (PMCD).</td>
</tr>
<tr>
<td>pmie</td>
<td>The Performance Metrics Inference Engine.</td>
</tr>
<tr>
<td>pmlogger</td>
<td>The performance metrics logger.</td>
</tr>
</tbody>
</table>
pmproxy | The realtime and historical performance metrics proxy, time series query and REST API service.

### 2.5. TOOLS DISTRIBUTED WITH PCP

The following table describes usage of various tools, which are distributed with PCP.

#### Table 2.2. Usage of tools distributed with PCP

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>pcp</strong></td>
<td>Displays the current status of a Performance Co-Pilot installation.</td>
</tr>
<tr>
<td><strong>pcp-atop</strong></td>
<td>Shows the system-level occupation of the most critical hardware resources from the performance point of view: CPU, memory, disk, and network.</td>
</tr>
<tr>
<td><strong>pcp-atopsar</strong></td>
<td>Generates a system-level activity report over a variety of system resource utilization. The report is generated from a raw logfile previously recorded using pmlogger or the -w option of pcp-atop.</td>
</tr>
<tr>
<td><strong>pcp-dmcache</strong></td>
<td>Displays information about configured Device Mapper Cache targets, such as: device IOPs, cache and metadata device utilization, as well as hit and miss rates and ratios for both reads and writes for each cache device.</td>
</tr>
<tr>
<td><strong>pcp-dstat</strong></td>
<td>Displays metrics of one system at a time. To display metrics of multiple systems, use <code>--host</code> option.</td>
</tr>
<tr>
<td><strong>pcp-free</strong></td>
<td>Reports on free and used memory in a system.</td>
</tr>
<tr>
<td><strong>pcp-htop</strong></td>
<td>Displays all processes running on a system along with their command line arguments in a manner similar to the <code>top</code> command, but allows you to scroll vertically and horizontally as well as interact using a mouse. You can also view processes in a tree format and select and act on multiple processes at once.</td>
</tr>
<tr>
<td><strong>pcp-ipcfs</strong></td>
<td>Displays information on the inter-process communication (IPC) facilities that the calling process has read access for.</td>
</tr>
<tr>
<td><strong>pcp-numastat</strong></td>
<td>Displays NUMA allocation statistics from the kernel memory allocator.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>pcp-pidstat</td>
<td>Displays information about individual tasks or processes running on the system such as: CPU percentage, memory and stack usage, scheduling, and priority. Reports live data for the local host by default.</td>
</tr>
<tr>
<td>pcp-ss</td>
<td>Displays socket statistics collected by the pmdasockets Performance Metrics Domain Agent (PMDA).</td>
</tr>
<tr>
<td>pcp-uptime</td>
<td>Displays how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.</td>
</tr>
<tr>
<td>pcp-vmstat</td>
<td>Provides a high-level system performance overview every 5 seconds. Displays information about processes, memory, paging, block IO, traps, and CPU activity.</td>
</tr>
<tr>
<td>pmchart</td>
<td>Plots performance metrics values available through the facilities of the Performance Co-Pilot.</td>
</tr>
<tr>
<td>pmclient</td>
<td>Displays high-level system performance metrics by using the Performance Metrics Application Programming Interface (PMAPI).</td>
</tr>
<tr>
<td>pmconfig</td>
<td>Displays the values of configuration parameters.ian</td>
</tr>
<tr>
<td>pmdbg</td>
<td>Displays available Performance Co-Pilot debug control flags and their values.</td>
</tr>
<tr>
<td>pmdiff</td>
<td>Compares the average values for every metric in either one or two archives, in a given time window, for changes that are likely to be of interest when searching for performance regressions.</td>
</tr>
<tr>
<td>pmdumplog</td>
<td>Displays control, metadata, index, and state information from a Performance Co-Pilot archive file.</td>
</tr>
<tr>
<td>pmdumptext</td>
<td>Outputs the values of performance metrics collected live or from a Performance Co-Pilot archive.</td>
</tr>
<tr>
<td>pmerr</td>
<td>Displays available Performance Co-Pilot error codes and their corresponding error messages.</td>
</tr>
<tr>
<td>pmfind</td>
<td>Finds PCP services on the network.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>pmie</td>
<td>An inference engine that periodically evaluates a set of arithmetic, logical, and rule expressions. The metrics are collected either from a live system, or from a Performance Co-Pilot archive file.</td>
</tr>
<tr>
<td>pmieconf</td>
<td>Displays or sets configurable pmie variables.</td>
</tr>
<tr>
<td>pmiectl</td>
<td>Manages non-primary instances of pmie.</td>
</tr>
<tr>
<td>pminfo</td>
<td>Displays information about performance metrics. The metrics are collected either from a live system, or from a Performance Co-Pilot archive file.</td>
</tr>
<tr>
<td>pmiostat</td>
<td>Reports I/O statistics for SCSI devices (by default) or device-mapper devices (with the -x dm option).</td>
</tr>
<tr>
<td>pmlc</td>
<td>Interactively configures active pmlogger instances.</td>
</tr>
<tr>
<td>pmlogcheck</td>
<td>Identifies invalid data in a Performance Co-Pilot archive file.</td>
</tr>
<tr>
<td>pmlogconf</td>
<td>Creates and modifies a pmlogger configuration file.</td>
</tr>
<tr>
<td>pmlogctl</td>
<td>Manages non-primary instances of pmlogger.</td>
</tr>
<tr>
<td>pmloglabel</td>
<td>Verifies, modifies, or repairs the label of a Performance Co-Pilot archive file.</td>
</tr>
<tr>
<td>pmlogsummary</td>
<td>Calculates statistical information about performance metrics stored in a Performance Co-Pilot archive file.</td>
</tr>
<tr>
<td>pmprobe</td>
<td>Determines the availability of performance metrics.</td>
</tr>
<tr>
<td>pmrep</td>
<td>Reports on selected, easily customizable, performance metrics values.</td>
</tr>
<tr>
<td>pmsocks</td>
<td>Allows access to a Performance Co-Pilot hosts through a firewall.</td>
</tr>
<tr>
<td>pmstat</td>
<td>Periodically displays a brief summary of system performance.</td>
</tr>
<tr>
<td>pmstore</td>
<td>Modifies the values of performance metrics.</td>
</tr>
<tr>
<td>pmtrace</td>
<td>Provides a command line interface to the trace PMDA.</td>
</tr>
<tr>
<td>pmval</td>
<td>Displays the current value of a performance metric.</td>
</tr>
</tbody>
</table>
2.6. PCP DEPLOYMENT ARCHITECTURES

Performance Co-Pilot (PCP) offers many options to accomplish advanced setups. From the huge variety of possible architectures, this section describes how to scale your PCP deployment based on the recommended deployment set up by Red Hat, sizing factors, and configuration options.

PCP supports multiple deployment architectures, based on the scale of the PCP deployment.

Available scaling deployment setup variants:

**Localhost**

Each service runs locally on the monitored machine. When you start a service without any configuration changes, this is the default deployment. Scaling beyond the individual node is not possible in this case.

By default, the deployment setup for Redis is standalone, localhost. However, Redis can optionally perform in a highly-available and highly scalable clustered fashion, where data is shared across multiple hosts. Another viable option is to deploy a Redis cluster in the cloud, or to utilize a managed Redis cluster from a cloud vendor.

**Decentralized**

The only difference between localhost and decentralized setup is the centralized Redis service. In this model, the host executes `pmlogger` service on each monitored host and retrieves metrics from a local `pmcd` instance. A local `pmproxy` service then exports the performance metrics to a central Redis instance.

**Centralized logging - pmlogger farm**

When the resource usage on the monitored hosts is constrained, another deployment option is a `pmlogger` farm, which is also known as centralized logging. In this setup, a single logger host executes multiple `pmlogger` processes, and each is configured to retrieve performance metrics from a different remote `pmcd` host. The centralized logger host is also configured to execute the `pmproxy` service, which discovers the resulting PCP archives logs and loads the metric data into a Redis instance.
Federated - multiple pmlogger farms

For large scale deployments, Red Hat recommends to deploy multiple pmlogger farms in a federated fashion. For example, one pmlogger farm per rack or data center. Each pmlogger farm loads the metrics into a central Redis instance.
NOTE

By default, the deployment setup for Redis is standalone, localhost. However, Redis can optionally perform in a highly-available and highly scalable clustered fashion, where data is shared across multiple hosts. Another viable option is to deploy a Redis cluster in the cloud, or to utilize a managed Redis cluster from a cloud vendor.
Additional resources

- `pcp(1)`, `pmlogger(1)`, `pmproxy(1)`, and `pmcd(1)` man pages
- Recommended deployment architecture

2.7. RECOMMENDED DEPLOYMENT ARCHITECTURE

The following table describes the recommended deployment architectures based on the number of monitored hosts.

Table 2.3. Recommended deployment architecture

<table>
<thead>
<tr>
<th>Number of hosts (N)</th>
<th>1-10</th>
<th>10-100</th>
<th>100-1000</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pmcd</code> servers</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td><code>pmlogger</code> servers</td>
<td>1 to N</td>
<td>N/10 to N</td>
<td>N/100 to N</td>
</tr>
<tr>
<td><code>pmproxy</code> servers</td>
<td>1 to N</td>
<td>1 to N</td>
<td>N/100 to N</td>
</tr>
<tr>
<td>Redis servers</td>
<td>1 to N</td>
<td>1 to N/10</td>
<td>N/100 to N/10</td>
</tr>
<tr>
<td>Redis cluster</td>
<td>No</td>
<td>Maybe</td>
<td>Yes</td>
</tr>
<tr>
<td>Recommended deployment setup</td>
<td>Localhost, Decentralized, or Centralized logging</td>
<td>Decentralized, Centralized logging, or Federated</td>
<td>Decentralized or Federated</td>
</tr>
</tbody>
</table>

2.8. SIZING FACTORS

The following are the sizing factors required for scaling:

Remote system size

The number of CPUs, disks, network interfaces, and other hardware resources affects the amount of data collected by each `pmlogger` on the centralized logging host.

Logged Metrics

The number and types of logged metrics play an important role. In particular, the per-process proc.* metrics require a large amount of disk space, for example, with the standard `pcp-zeroconf` setup, 10s logging interval, 11 MB without proc metrics versus 155 MB with proc metrics - a factor of 10 times more. Additionally, the number of instances for each metric, for example the number of CPUs, block devices, and network interfaces also impacts the required storage capacity.

Logging Interval

The interval how often metrics are logged, affects the storage requirements. The expected daily PCP archive file sizes are written to the `pmlogger.log` file for each `pmlogger` instance. These values are uncompressed estimates. Since PCP archives compress very well, approximately 10:1, the actual long term disk space requirements can be determined for a particular site.

`pmlogrewrite`
After every PCP upgrade, the **pmlogrewrite** tool is executed and rewrites old archives if there were changes in the metric metadata from the previous version and the new version of PCP. This process duration scales linear with the number of archives stored.

### Additional resources

- **pmlogrewrite(1)** and **pmlogger(1)** man pages

## 2.9. CONFIGURATION OPTIONS FOR PCP SCALING

The following are the configuration options, which are required for scaling:

### sysctl and rlimit settings

When archive discovery is enabled, **pmproxy** requires four descriptors for every **pmlogger** that it is monitoring or log-tailing, along with the additional file descriptors for the service logs and **pmproxy** client sockets, if any. Each **pmlogger** process uses about 20 file descriptors for the remote **pmcd** socket, archive files, service logs, and others. In total, this can exceed the default 1024 soft limit on a system running around 200 **pmlogger** processes. The **pmproxy** service in **pcp-5.3.0** and later automatically increases the soft limit to the hard limit. On earlier versions of PCP, tuning is required if a high number of **pmlogger** processes are to be deployed, and this can be accomplished by increasing the soft or hard limits for **pmlogger**. For more information, see [How to set limits (ulimit)] for services run by systemd.

### Local Archives

The **pmlogger** service stores metrics of local and remote **pmcds** in the `/var/log/pcp/pmlogger/` directory. To control the logging interval of the local system, update the `/etc/pcp/pmlogger/control.d/configfile` file and add `-t X` in the arguments, where `X` is the logging interval in seconds. To configure which metrics should be logged, execute `pmlogconf /var/lib/pcp/config/pmlogger/config.clienthostname`. This command deploys a configuration file with a default set of metrics, which can optionally be further customized. To specify retention settings, that is when to purge old PCP archives, update the `/etc/sysconfig/pmlogger_timers` file and specify `PMLOGGER_DAILY_PARAMS="-E -k X"`, where `X` is the amount of days to keep PCP archives.

### Redis

The **pmproxy** service sends logged metrics from **pmlogger** to a Redis instance. The following are the available two options to specify the retention settings in the `/etc/pcp/pmproxy/pmproxy.conf` configuration file:

- **stream.expire** specifies the duration when stale metrics should be removed, that is metrics which were not updated in a specified amount of time in seconds.

- **stream.maxlen** specifies the maximum number of metric values for one metric per host. This setting should be the retention time divided by the logging interval, for example 20160 for 14 days of retention and 60s logging interval (60*60*24*14/60)

### Additional resources

- **pmproxy(1)**, **pmlogger(1)**, and **sysctl(8)** man pages

## 2.10. EXAMPLE: ANALYZING THE CENTRALIZED LOGGING DEPLOYMENT

The following results were gathered on a centralized logging setup, also known as **pmlogger farm**
The following results were gathered on a centralized logging setup, also known as pmlogger farm deployment, with a default `pcp-zeroconf 5.3.0` installation, where each remote host is an identical container instance running `pmcd` on a server with 64 CPU cores, 376 GB RAM, and one disk attached.

The logging interval is 10s, proc metrics of remote nodes are not included, and the memory values refer to the Resident Set Size (RSS) value.

Table 2.4. Detailed utilization statistics for 10s logging interval

<table>
<thead>
<tr>
<th>Number of Hosts</th>
<th>10</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCP Archives Storage per Day</td>
<td>91 MB</td>
<td>522 MB</td>
</tr>
<tr>
<td><strong>pmlogger</strong> Memory</td>
<td>160 MB</td>
<td>580 MB</td>
</tr>
<tr>
<td><strong>pmlogger</strong> Network per Day (In)</td>
<td>2 MB</td>
<td>9 MB</td>
</tr>
<tr>
<td><strong>pmproxy</strong> Memory</td>
<td>1.4 GB</td>
<td>6.3 GB</td>
</tr>
<tr>
<td>Redis Memory per Day</td>
<td>2.6 GB</td>
<td>12 GB</td>
</tr>
</tbody>
</table>

Table 2.5. Used resources depending on monitored hosts for 60s logging interval

<table>
<thead>
<tr>
<th>Number of Hosts</th>
<th>10</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCP Archives Storage per Day</td>
<td>20 MB</td>
<td>120 MB</td>
<td>271 MB</td>
</tr>
<tr>
<td><strong>pmlogger</strong> Memory</td>
<td>104 MB</td>
<td>524 MB</td>
<td>1049 MB</td>
</tr>
<tr>
<td><strong>pmlogger</strong> Network per Day (In)</td>
<td>0.38 MB</td>
<td>1.75 MB</td>
<td>3.48 MB</td>
</tr>
<tr>
<td><strong>pmproxy</strong> Memory</td>
<td>2.67 GB</td>
<td>5.5 GB</td>
<td>9 GB</td>
</tr>
<tr>
<td>Redis Memory per Day</td>
<td>0.54 GB</td>
<td>2.65 GB</td>
<td>5.3 GB</td>
</tr>
</tbody>
</table>

**NOTE**

The **pmproxy** queues Redis requests and employs Redis pipelining to speed up Redis queries. This can result in high memory usage. For troubleshooting this issue, see [Troubleshooting high memory usage](#).

2.11. EXAMPLE: ANALYZING THE FEDERATED SETUP DEPLOYMENT

The following results were observed on a federated setup, also known as multiple pmlogger farms, consisting of three centralized logging (pmlogger farm) setups, where each pmlogger farm was monitoring 100 remote hosts, that is 300 hosts in total.
This setup of the pmlogger farms is identical to the configuration mentioned in the Example: Analyzing the centralized logging deployment for 60s logging interval, except that the Redis servers were operating in cluster mode.

### Table 2.6. Used resources depending on federated hosts for 60s logging interval

<table>
<thead>
<tr>
<th>PCP Archives Storage per Day</th>
<th>pmlogger Memory</th>
<th>Network per Day (In/Out)</th>
<th>pmproxy Memory</th>
<th>Redis Memory per Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>277 MB</td>
<td>1058 MB</td>
<td>15.6 MB / 12.3 MB</td>
<td>6-8 GB</td>
<td>5.5 GB</td>
</tr>
</tbody>
</table>

Here, all values are per host. The network bandwidth is higher due to the inter-node communication of the Redis cluster.

### 2.12. TROUBLESHOOTING HIGH MEMORY USAGE

The following scenarios can result in high memory usage:

- The pmproxy process is busy processing new PCP archives and does not have spare CPU cycles to process Redis requests and responses.
- The Redis node or cluster is overloaded and cannot process incoming requests on time.

The pmproxy service daemon uses Redis streams and supports the configuration parameters, which are PCP tuning parameters and affects Redis memory usage and key retention. The `/etc/pcp/pmproxy/pmproxy.conf` file lists the available configuration options for pmproxy and the associated APIs.

This section describes how to troubleshoot high memory usage issue.

**Prerequisites**

1. Install the `pcp-pmda-redis` package:

   ```bash
   # yum install pcp-pmda-redis
   ```

2. Install the redis PMDA:

   ```bash
   # cd /var/lib/pcp/pmdas/redis && ./Install
   ```

**Procedure**

- To troubleshoot high memory usage, execute the following command and observe the `inflight` column:

  ```bash
  $ pmrep :pmproxy
  backlog inflight reqs/s resp/s wait req err resp err changed throttled
  byte count count/s count/s s/s count/s count/s count/s count/s
  14:59:08 0     0     N/A    N/A    N/A    N/A    N/A    N/A    N/A
  14:59:09 0     0     2268.9 2268.9 28     0     0     2.0    4.0
  14:59:10 0     0     0.0    0.0    0      0     0     0.0    0.0
  14:59:11 0     0     0.0    0.0    0      0     0     0.0    0.0
  ```

This column shows how many Redis requests are in-flight, which means they are queued or sent.
This column shows how many Redis requests are in-flight, which means they are queued or sent, and no reply was received so far.

A high number indicates one of the following conditions:

- The `pmproxy` process is busy processing new PCP archives and does not have spare CPU cycles to process Redis requests and responses.

- The Redis node or cluster is overloaded and cannot process incoming requests on time.

- To troubleshoot the high memory usage issue, reduce the number of `pmlogger` processes for this farm, and add another pmlogger farm. Use the federated - multiple pmlogger farms setup. If the Redis node is using 100% CPU for an extended amount of time, move it to a host with better performance or use a clustered Redis setup instead.

- To view the `pmproxy.redis.*` metrics, use the following command:

  ```bash
  $ pminfo -ftd pmproxy.redis
  pmproxy.redis.responses.wait [wait time for responses]
  Data Type: 64-bit unsigned int  InDom: PM_INDOM_NULL 0xffffffff
  Semantics: counter  Units: microsec
  value 546028367374
  pmproxy.redis.responses.error [number of error responses]
  Data Type: 64-bit unsigned int  InDom: PM_INDOM_NULL 0xffffffff
  Semantics: counter  Units: count
  value 1164
  ...
  pmproxy.redis.requests.inflight.bytes [bytes allocated for inflight requests]
  Data Type: 64-bit int  InDom: PM_INDOM_NULL 0xffffffff
  Semantics: discrete  Units: byte
  value 0
  pmproxy.redis.requests.inflight.total [inflight requests]
  Data Type: 64-bit unsigned int  InDom: PM_INDOM_NULL 0xffffffff
  Semantics: discrete  Units: count
  value 0
  ...
  ``

To view how many Redis requests are inflight, see the `pmproxy.redis.requests.inflight.total` metric and `pmproxy.redis.requests.inflight.bytes` metric to view how many bytes are occupied by all current inflight Redis requests.

In general, the redis request queue would be zero but can build up based on the usage of large pmlogger farms, which limits scalability and can cause high latency for `pmproxy` clients.

- Use the `pminfo` command to view information about performance metrics. For example, to view the `redis.*` metrics, use the following command:

  ```bash
  $ pminfo -ftd redis
  redis.redis_build_id [Build ID]
  Data Type: string  InDom: 24.0 0x6000000
  Semantics: discrete  Units: count
  value "87e335e57cffa755"
  redis.total_commands_processed [Total number of commands processed by the server]
  Data Type: 64-bit unsigned int  InDom: 24.0 0x6000000
  Semantics: counter  Units: count
  ```
To view the peak memory usage, see the `redis.used_memory_peak` metric.

Additional resources

- `pmdaredis(1)`, `pmproxy(1)`, and `pminfo(1)` man pages
- PCP deployment architectures
CHAPTER 3. LOGGING PERFORMANCE DATA WITH PMLOGGER

With the PCP tool you can log the performance metric values and replay them later. This allows you to perform a retrospective performance analysis.

Using the `pmlogger` tool, you can:

- Create the archived logs of selected metrics on the system
- Specify which metrics are recorded on the system and how often

3.1. MODIFYING THE PMLOGGER CONFIGURATION FILE WITH PMLOGCONF

When the `pmlogger` service is running, PCP logs a default set of metrics on the host.

Use the `pmlogconf` utility to check the default configuration. If the `pmlogger` configuration file does not exist, `pmlogconf` creates it with a default metric values.

Prerequisites

- PCP is installed. For more information, see Installing and enabling PCP.

Procedure

1. Create or modify the `pmlogger` configuration file:

   ```bash
   # pmlogconf -r /var/lib/pcp/config/pmlogger/config.default
   ```

2. Follow `pmlogconf` prompts to enable or disable groups of related performance metrics and to control the logging interval for each enabled group.

Additional resources

- `pmlogconf(1)` and `pmlogger(1)` man pages
- Tools distributed with PCP
- System services distributed with PCP

3.2. EDITING THE PMLOGGER CONFIGURATION FILE MANUALLY

To create a tailored logging configuration with specific metrics and given intervals, edit the `pmlogger` configuration file manually. The default `pmlogger` configuration file is `/var/lib/pcp/config/pmlogger/config.default`. The configuration file specifies which metrics are logged by the primary logging instance.

In manual configuration, you can:

- Record metrics which are not listed in the automatic configuration.
- Choose custom logging frequencies.
• Add PMDA with the application metrics.

Prerequisites

• PCP is installed. For more information, see Installing and enabling PCP.

Procedure

• Open and edit the `/var/lib/pcp/config/pmlogger/config.default` file to add specific metrics:

```bash
# It is safe to make additions from here on ...
#
log mandatory on every 5 seconds {
    xfs.write
    xfs.write_bytes
    xfs.read
    xfs.read_bytes
}
log mandatory on every 10 seconds {
    xfs.allocs
    xfs.block_map
    xfs.transactions
    xfs.log
}

[access]
disallow * : all;
allow localhost : enquire;
```

Additional resources

• pmlogger(1) man page

• Tools distributed with PCP

• System services distributed with PCP

3.3. ENABLING THE PMLOGGER SERVICE

The pmlogger service must be started and enabled to log the metric values on the local machine.

This procedure describes how to enable the pmlogger service.

Prerequisites

• PCP is installed. For more information, see Installing and enabling PCP.

Procedure

• Start and enable the pmlogger service:
# systemctl start pmlogger
# systemctl enable pmlogger

Verification steps

- Verify if the **pmlogger** service is enabled:

```
# pcp
```

Performance Co-Pilot configuration on workstation:

```
platform: Linux workstation 4.18.0-80.el8.x86_64 #1 SMP Wed Mar 13 12:02:46 UTC 2019
x86_64
hardware: 12 cpus, 2 disks, 1 node, 36023MB RAM
timezone: CEST-2
services: pmcd
pmcd: Version 4.3.0-1, 8 agents, 1 client
pmda: root pmcd proc xfs linux mmv kvm jbd2
pmlogger: primary logger: /var/log/pcp/pmlogger/workstation/20190827.15.54
```

Additional resources

- **pmlogger(1)** man page
- Tools distributed with PCP
- System services distributed with PCP
- `/var/lib/pcp/config/pmlogger/config.default` file

### 3.4. SETTING UP A CLIENT SYSTEM FOR METRICS COLLECTION

This procedure describes how to set up a client system so that a central server can collect metrics from clients running PCP.

**Prerequisites**

- PCP is installed. For more information, see [Installing and enabling PCP](#).

**Procedure**

1. Install the **pcp-system-tools** package:

   ```
   # yum install pcp-system-tools
   ```

2. Configure an IP address for **pmcd**:

   ```
   # echo "-i 192.168.4.62" >>/etc/pcp/pmcd/pmcd.options
   ```

   Replace `192.168.4.62` with the IP address, the client should listen on.

   By default, **pmcd** is listening on the localhost.
3. Configure the firewall to add the public zone permanently:

```
# firewall-cmd --permanent --zone=public --add-port=44321/tcp
success

# firewall-cmd --reload
success
```

4. Set an SELinux boolean:

```
# setsebool -P pcp_bind_all_unreserved_ports on
```

5. Enable the pmcd and pmlogger services:

```
# systemctl enable pmcd pmlogger
# systemctl restart pmcd pmlogger
```

**Verification steps**

- Verify if the pmcd is correctly listening on the configured IP address:

```
# ss -tlp | grep 44321
LISTEN  0  5  127.0.0.1:44321  0.0.0.0:*  users:("pmcd",pid=151595,fd=6))
LISTEN  0  5  192.168.4.62:44321  0.0.0.0:*  users:("pmcd",pid=151595,fd=0))
LISTEN  0  5  [::1]:44321  [::]:*  users:("pmcd",pid=151595,fd=7))
```

**Additional resources**

- pmlogger(1), firewall-cmd(1), ss(8), and setsebool(8) man pages
- Tools distributed with PCP
- System services distributed with PCP
- /var/lib/pcp/config/pmlogger/config.default file

**3.5. SETTING UP A CENTRAL SERVER TO COLLECT DATA**

This procedure describes how to create a central server to collect metrics from clients running PCP.

**Prerequisites**

- PCP is installed. For more information, see [Installing and enabling PCP](#).
- Client is configured for metrics collection. For more information, see [Setting up a client system for metrics collection](#).

**Procedure**

1. Install the pcp-system-tools package:

```bash
# yum install pcp-system-tools
```
2. Create the /etc/pcp/pmlogger/control.d/remote file with the following content:

```bash
# DO NOT REMOVE OR EDIT THE FOLLOWING LINE
$version=1.1

192.168.4.13 n n PCP_LOG_DIR/pmlogger/rhel7u4a -r -T24h10m -c config.rhel7u4a
192.168.4.14 n n PCP_LOG_DIR/pmlogger/rhel6u10a -r -T24h10m -c config.rhel6u10a
192.168.4.62 n n PCP_LOG_DIR/pmlogger/rhel8u1a -r -T24h10m -c config.rhel8u1a
```

Replace 192.168.4.13, 192.168.4.14, and 192.168.4.62 with the client IP addresses.

3. Enable the **pmcd** and **pmlogger** services:

```bash
# systemctl enable pmcd pmlogger
# systemctl restart pmcd pmlogger
```

**Verification steps**

- Ensure that you can access the latest archive file from each directory:

  ```bash
  # for i in /var/log/pcp/pmlogger/rhel*//*.0; do pmdumplog -L $i; done
  Log Label (Log Format Version 2)
  Performance metrics from host rhel6u10a.local
  ending     Mon Nov 25 22:06:04.874 2019
  Archive timezone: JST-9
  PID for pmlogger: 24002
  PID for pmlogger: 24002
  Log Label (Log Format Version 2)
  Performance metrics from host rhel7u4a
  commencing Tue Nov 26 06:49:24.954 2019
  ending     Tue Nov 26 07:06:24.979 2019
  Archive timezone: CET-1
  PID for pmlogger: 10941
  [...]
  ```

The archive files from the /var/log/pcp/pmlogger/ directory can be used for further analysis and graphing.

**Additional resources**

- **pmlogger(1)** man page
- Tools distributed with PCP
- System services distributed with PCP
- /var/lib/pcp/config/pmlogger/config.default file

### 3.6. REPLAYING THE PCP LOG ARCHIVES WITH PMDUMPTEXT

After recording the metric data, you can replay the PCP log archives. To export the logs to text files and import them into spreadsheets, use PCP utilities such as **pmdumptext**, **pmrep**, or **pmlogsummary**.

Using the **pmdumptext** tool, you can:
• View the log files
• Parse the selected PCP log archive and export the values into an ASCII table
• Extract the entire archive log or only select metric values from the log by specifying individual metrics on the command line

Prerequisites
• PCP is installed. For more information, see Installing and enabling PCP.
• The pmlogger service is enabled. For more information, see Enabling the pmlogger service.
• Install the pcp-gui package:

  # yum install pcp-gui

Procedure
• Display the data on the metric:

  $ pmdumptext -t 5seconds -H -a 20170605 xfs.perdev.log.writes

  Time local::xfs.perdev.log.writes["/dev/mapper/fedora-home"]
  local::xfs.perdev.log.writes["/dev/mapper/fedora-root"]
  ? 0.000 0.000
  none count / second count / second
  Mon Jun 5 12:28:45 ??
  Mon Jun 5 12:28:50 0.000 0.000
  Mon Jun 5 12:28:55 0.200 0.200
  Mon Jun 5 12:29:00 6.800 1.000

  The mentioned example displays the data on the xfs.perdev.log metric collected in an archive at a 5 second interval and display all the headers.

  NOTE

  Replace 20170605 in this example with a filename containing the pmlogger archive you want to display data for.

Additional resources
• pmlogger(1), pmdumptext(1), pmrep(1), and pmlogsummary(1) man pages
• Tools distributed with PCP
• System services distributed with PCP
CHAPTER 4. MONITORING PERFORMANCE WITH PERFORMANCE CO-PILOT

Performance Co-Pilot (PCP) is a suite of tools, services, and libraries for monitoring, visualizing, storing, and analyzing system-level performance measurements.

As a system administrator, you can monitor the system’s performance using the the PCP application in Red Hat Enterprise Linux 9.

4.1. MONITORING POSTFIX WITH PMDA-POSTFIX

This procedure describes how to monitor performance metrics of the postfix mail server with pmda-postfix. It helps to check how many emails are received per second.

Prerequisites

- PCP is installed. For more information, see Installing and enabling PCP.
- The pmlogger service is enabled. For more information, see Enabling the pmlogger service.

Procedure

1. Install the following packages:
   a. Install the pcp-system-tools:
      ```
      # yum install pcp-system-tools
      ```
   b. Install the pmda-postfix package to monitor postfix:
      ```
      # yum install pcp-pmda-postfix postfix
      ```
   c. Install the logging daemon:
      ```
      # yum install rsyslog
      ```
   d. Install the mail client for testing:
      ```
      # yum install mutt
      ```

2. Enable the postfix and rsyslog services:

   ```
   # systemctl enable postfix rsyslog
   # systemctl restart postfix rsyslog
   ```

3. Enable the SELinux boolean, so that pmda-postfix can access the required log files:

   ```
   # setsebool -P pcp_read_generic_logs=on
   ```

4. Install the PMDA:

   ```
   # cd /var/lib/pcp/pmdas/postfix/
   ```
# Install

Updating the Performance Metrics Name Space (PMNS) ...
Terminate PMDA if already installed ...
Updating the PMCD control file, and notifying PMCD ...
Waiting for pmcd to terminate ...
Starting pmcd ...
Check postfix metrics have appeared ... 7 metrics and 58 values

Verification steps

- Verify the `pmda-postfix` operation:
  ```
  echo testmail | mutt root
  ```

- Verify the available metrics:
  ```
  # pminfo postfix
  ```
  ```
  postfix.received
  postfix.sent
  postfix.queues.incoming
  postfix.queues.maildrop
  postfix.queues.hold
  postfix.queues.deferred
  postfix.queues.active
  ```

Additional resources

- `rsyslogd(8)`, `postfix(1)`, and `setsebool(8)` man pages
- Tools distributed with PCP
- System services distributed with PCP
- `/var/lib/pcp/config/pmlogger/config.default` file

4.2. VISUALLY TRACING PCP LOG ARCHIVES WITH THE PCP CHARTS APPLICATION

After recording metric data, you can replay the PCP log archives as graphs. The metrics are sourced from one or more live hosts with alternative options to use metric data from PCP log archives as a source of historical data. To customize the PCP Charts application interface to display the data from the performance metrics, you can use line plot, bar graphs, or utilization graphs.

Using the PCP Charts application, you can:

- Replay the data in the PCP Charts application application and use graphs to visualize the retrospective data alongside live data of the system.
- Plot performance metric values into graphs.
- Display multiple charts simultaneously.
Prerequisites

- PCP is installed. For more information, see Installing and enabling PCP.
- Logged performance data with the pmlogger. For more information, see Logging performance data with pmlogger.
- Install the pcp-gui package:
  ```
  # yum install pcp-gui
  ```

Procedure

1. Launch the PCP Charts application from the command line:
  ```
  # pmchart
  ```

   Figure 4.1. PCP Charts application

   ![PCP Charts application](image)

   The `pmtime` server settings are located at the bottom. The **start** and **pause** button allows you to control:
   - The interval in which PCP polls the metric data
   - The date and time for the metrics of historical data

2. Click **File** and then **New Chart** to select metric from both the local machine and remote machines by specifying their host name or address. Advanced configuration options include the ability to manually set the axis values for the chart, and to manually choose the color of the plots.

3. Record the views created in the PCP Charts application:
   Following are the options to take images or record the views created in the PCP Charts application:
   - Click **File** and then **Export** to save an image of the current view.
• Click Record and then Start to start a recording. Click Record and then Stop to stop the recording. After stopping the recording, the recorded metrics are archived to be viewed later.

4. Optional: In the PCP Charts application, the main configuration file, known as the view, allows the metadata associated with one or more charts to be saved. This metadata describes all chart aspects, including the metrics used and the chart columns. Save the custom view configuration by clicking File and then Save View, and load the view configuration later. The following example of the PCP Charts application view configuration file describes a stacking chart graph showing the total number of bytes read and written to the given XFS file system loop1:

```
#kmchart
version 1

chart title "Filesystem Throughput /loop1" style stacking antialiasing off
plot legend "Read rate" metric xfs.read_bytes instance "loop1"
plot legend "Write rate" metric xfs.write_bytes instance "loop1"
```

Additional resources

• pmchart(1) and pmtime(1) man pages

• Tools distributed with PCP

4.3. COLLECTING DATA FROM SQL SERVER USING PCP

The SQL Server agent is available in Performance Co-Pilot (PCP), which helps you to monitor and analyze database performance issues.

This procedure describes how to collect data for Microsoft SQL Server via pcp on your system.

Prerequisites

• You have installed Microsoft SQL Server for Red Hat Enterprise Linux and established a ‘trusted’ connection to an SQL server.

• You have installed the Microsoft ODBC driver for SQL Server for Red Hat Enterprise Linux.

Procedure

1. Install PCP:
   ```
   # yum install pcp-zeroconf
   ```

2. Install packages required for the pyodbc driver:
   ```
   # yum install python3-pyodbc
   ```

3. Install the mssql agent:
   a. Install the Microsoft SQL Server domain agent for PCP:
      ```
      # yum install pcp-pmda-mssql
      ```
b. Edit the `/etc/pcp/mssql/mssql.conf` file to configure the SQL server account's username and password for the `mssql` agent. Ensure that the account you configure has access rights to performance data.

```plaintext
username: user_name
password: user_password
```

Replace `user_name` with the SQL Server account and `user_password` with the SQL Server user password for this account.

4. Install the agent:

```bash
# cd /var/lib/pcp/pmdas/mssql
# ./Install
Updating the Performance Metrics Name Space (PMNS) ...
Terminate PMDA if already installed ...
Updating the PMCD control file, and notifying PMCD ...
Check mssql metrics have appeared ... 168 metrics and 598 values
[...]
```

**Verification steps**

- Using the `pcp` command, verify if the SQL Server PMDA (`mssql`) is loaded and running:

```bash
$ pcp
Performance Co-Pilot configuration on rhel.local:

platform: Linux rhel.local 4.18.0-167.el8.x86_64 #1 SMP Sun Dec 15 01:24:23 UTC 2019 x86_64
hardware: 2 cpus, 1 disk, 1 node, 2770MB RAM
timezone: PDT+7
services: pmcd pmproxy
  pmcd: Version 5.0.2-1, 12 agents, 4 clients
  pmda: root pmcd proc pmproxy xfs linux nfsclient mmv kvm mssql
  jbd2 dm
pmlogger: primary logger: /var/log/pcp/pmlogger/rhel.local/20200326.16.31
pmie: primary engine: /var/log/pcp/pmie/rhel.local/pmie.log

```

- View the complete list of metrics that PCP can collect from the SQL Server:

```bash
# pminfo mssql
```

- After viewing the list of metrics, you can report the rate of transactions. For example, to report on the overall transaction count per second, over a five second time window:

```bash
# pmval -t 1 -T 5 mssql.databases.transactions
```

- View the graphical chart of these metrics on your system by using the `pmchart` command. For more information, see [Visually tracing PCP log archives with the PCP Charts application](https://example.com).

**Additional resources**

- `pcp(1)`, `pminfo(1)`, `pmval(1)`, `pmchart(1)`, and `pmdamssql(1)` man pages
• Performance Co-Pilot for Microsoft SQL Server with RHEL 8.2 Red Hat Developers Blog post
CHAPTER 5. PERFORMANCE ANALYSIS OF XFS WITH PCP

The XFS PMDA ships as part of the pcp package and is enabled by default during the installation. It is used to gather performance metric data of XFS file systems in Performance Co-Pilot (PCP).

This section describes how to analyze XFS file system's performance using PCP.

5.1. INSTALLING XFS PMDA MANUALLY

If the XFS PMDA is not listed in the pcp configuration output, install the PMDA agent manually.

This procedure describes how to manually install the PMDA agent.

Prerequisites

- PCP is installed. For more information, see Installing and enabling PCP.

Procedure

1. Navigate to the xfs directory:

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

Verification steps

- Verify that the pmcd process is running on the host and the XFS PMDA is listed as enabled in the configuration:

   ```
   # pcp
   Performance Co-Pilot configuration on workstation:
   platform: Linux workstation 4.18.0-80.el8.x86_64 #1 SMP Wed Mar 13 12:02:46 UTC 2019 x86_64
   hardware: 12 cpus, 2 disks, 1 node, 36023MB RAM
   timezone: CEST-2
   services: pmcd
   pmcd: Version 4.3.0-1, 8 agents
   pmda: root pmcd proc xfs linux mmv kvm jbd2
   ```

Additional resources

- pmcd(1) man page
- Tools distributed with PCP

5.2. EXAMINING XFS PERFORMANCE METRICS WITH PMINFO

PCP enables XFS PMDA to allow the reporting of certain XFS metrics per each of the mounted XFS file systems. This makes it easier to pinpoint specific mounted file system issues and evaluate performance.

The pminfo command provides per-device XFS metrics for each mounted XFS file system.
This procedure displays a list of all available metrics provided by the XFS PMDA.

Prerequisites

- PCP is installed. For more information, see Installing and enabling PCP.

Procedure

- Display the list of all available metrics provided by the XFS PMDA:
  
  ```
  # pminfo xfs
  ```

- Display information for the individual metrics. The following examples examine specific XFS read and write metrics using the `pminfo` tool:
  - Display a short description of the `xfs.write_bytes` metric:
    
    ```
    # pminfo --oneline xfs.write_bytes
    xfs.write_bytes [number of bytes written in XFS file system write operations]
    ```
  - Display a long description of the `xfs.read_bytes` metric:
    
    ```
    # pminfo --helptext xfs.read_bytes
    xfs.read_bytes
    Help:
    This is the number of bytes read via read(2) system calls to files in XFS file systems. It can be used in conjunction with the read_calls count to calculate the average size of the read operations to file in XFS file systems.
    ```
  - Obtain the current performance value of the `xfs.read_bytes` metric:
    
    ```
    # pminfo --fetch xfs.read_bytes
    xfs.read_bytes
    value 4891346238
    ```
  - Obtain per-device XFS metrics with `pminfo`:
    
    ```
    # pminfo --fetch --oneline xfs.perdev.read xfs.perdev.write
    xfs.perdev.read [number of XFS file system read operations]
    inst [0 or "loop1"] value 0
    inst [0 or "loop2"] value 0
    xfs.perdev.write [number of XFS file system write operations]
    inst [0 or "loop1"] value 86
    inst [0 or "loop2"] value 0
    ```

Additional resources
5.3. RESETTING XFS PERFORMANCE METRICS WITH PMSTORE

With PCP, you can modify the values of certain metrics, especially if the metric acts as a control variable, such as the `xfs.control.reset` metric. To modify a metric value, use the `pmstore` tool.

This procedure describes how to reset XFS metrics using the `pmstore` tool.

Prerequisites

- PCP is installed. For more information, see Installing and enabling PCP.

Procedure

1. Display the value of a metric:

   ```
   $ pminfo -f xfs.write
   xfs.write
   value 325262
   ```

2. Reset all the XFS metrics:

   ```
   # pmstore xfs.control.reset 1
   xfs.control.reset old value=0 new value=1
   ```

Verification steps

- View the information after resetting the metric:

  ```
  $ pminfo --fetch xfs.write
  xfs.write
  value 0
  ```

Additional resources

- `pmstore(1)` and `pminfo(1)` man pages
- Tools distributed with PCP
- PCP metric groups for XFS

5.4. PCP METRIC GROUPS FOR XFS

The following table describes the available PCP metric groups for XFS.
<table>
<thead>
<tr>
<th>Metric Group</th>
<th>Metrics provided</th>
</tr>
</thead>
<tbody>
<tr>
<td>xfs.*</td>
<td>General XFS metrics including the read and write operation counts, read and write byte counts. Along with counters for the number of times inodes are flushed, clustered and number of failure to cluster.</td>
</tr>
<tr>
<td>xfs.allocs.*</td>
<td>Range of metrics regarding the allocation of objects in the file system, these include number of extent and block creations/frees. Allocation tree lookup and compares along with extend record creation and deletion from the btree.</td>
</tr>
<tr>
<td>xfs.alloc_btree.*</td>
<td></td>
</tr>
<tr>
<td>xfs.block_map.*</td>
<td>Metrics include the number of block map read/write and block deletions, extent list operations for insertion, deletions and lookups. Also operations counters for compares, lookups, insertions and deletion operations from the blockmap.</td>
</tr>
<tr>
<td>xfs.bmap_btree.*</td>
<td></td>
</tr>
<tr>
<td>xfs.dir_ops.*</td>
<td>Counters for directory operations on XFS file systems for creation, entry deletions, count of “getdent” operations.</td>
</tr>
<tr>
<td>xfs.transactions.*</td>
<td>Counters for the number of meta-data transactions, these include the count for the number of synchronous and asynchronous transactions along with the number of empty transactions.</td>
</tr>
<tr>
<td>xfs.inode_ops.*</td>
<td>Counters for the number of times that the operating system looked for an XFS inode in the inode cache with different outcomes. These count cache hits, cache misses, and so on.</td>
</tr>
<tr>
<td>xfs.log.*</td>
<td>Counters for the number of log buffer writes over XFS file systems includes the number of blocks written to disk. Metrics also for the number of log flushes and pinning.</td>
</tr>
<tr>
<td>xfs.log_tail.*</td>
<td></td>
</tr>
<tr>
<td>xfs.xstrat.*</td>
<td>Counts for the number of bytes of file data flushed out by the XFS flush deamon along with counters for number of buffers flushed to contiguous and non-contiguous space on disk.</td>
</tr>
<tr>
<td>xfs.attr.*</td>
<td>Counts for the number of attribute get, set, remove and list operations over all XFS file systems.</td>
</tr>
<tr>
<td>xfs.quota.*</td>
<td>Metrics for quota operation over XFS file systems, these include counters for number of quota reclaims, quota cache misses, cache hits and quota data reclaims.</td>
</tr>
</tbody>
</table>
5.5. PER-DEVICE PCP METRIC GROUPS FOR XFS

The following table describes the available per-device PCP metric group for XFS.

<table>
<thead>
<tr>
<th>Metric Group</th>
<th>Metrics provided</th>
</tr>
</thead>
<tbody>
<tr>
<td>xfs.buffer.*</td>
<td>Range of metrics regarding XFS buffer objects. Counters include the number of requested buffer calls, successful buffer locks, waited buffer locks, miss_locks, miss_retries and buffer hits when looking up pages.</td>
</tr>
<tr>
<td>xfs.btree.*</td>
<td>Metrics regarding the operations of the XFS btree.</td>
</tr>
<tr>
<td>xfs.control.reset</td>
<td>Configuration metrics which are used to reset the metric counters for the XFS stats. Control metrics are toggled by means of the pmstore tool.</td>
</tr>
</tbody>
</table>

Table 5.2. Per-device PCP metric groups for XFS

<table>
<thead>
<tr>
<th>Metric Group</th>
<th>Metrics provided</th>
</tr>
</thead>
<tbody>
<tr>
<td>xfs.perdev.*</td>
<td>General XFS metrics including the read and write operation counts, read and write byte counts. Along with counters for the number of times inodes are flushed, clustered and number of failure to cluster.</td>
</tr>
<tr>
<td>xfs.perdev.allocs.*</td>
<td>Range of metrics regarding the allocation of objects in the file system, these include number of extent and block creations/frees. Allocation tree lookup and compares along with extend record creation and deletion from the btree.</td>
</tr>
<tr>
<td>xfs.perdev.alloc_btree.*</td>
<td></td>
</tr>
<tr>
<td>xfs.perdev.block_map.*</td>
<td>Metrics include the number of block map read/write and block deletions, extent list operations for insertion, deletions and lookups. Also operations counters for compares, lookups, insertions and deletion operations from the blockmap.</td>
</tr>
<tr>
<td>xfs.perdev.bmap_btree.*</td>
<td></td>
</tr>
<tr>
<td>xfs.perdev.dir_ops.*</td>
<td>Counters for directory operations of XFS file systems for creation, entry deletions, count of “getdent” operations.</td>
</tr>
<tr>
<td>xfs.perdev.transactions.*</td>
<td>Counters for the number of meta-data transactions, these include the count for the number of synchronous and asynchronous transactions along with the number of empty transactions.</td>
</tr>
<tr>
<td>xfs.perdev.inode_ops.*</td>
<td>Counters for the number of times that the operating system looked for an XFS inode in the inode cache with different outcomes. These count cache hits, cache misses, and so on.</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>xfs.perdev.log.*</code></td>
<td>Counters for the number of log buffer writes over XFS filesystems includes the number of blocks written to disk. Metrics also for the number of log flushes and pinning.</td>
</tr>
<tr>
<td><code>xfs.perdev.log_tail.*</code></td>
<td></td>
</tr>
<tr>
<td><code>xfs.perdev.xstrat.*</code></td>
<td>Counts for the number of bytes of file data flushed out by the XFS flush daemon along with counters for number of buffers flushed to contiguous and non-contiguous space on disk.</td>
</tr>
<tr>
<td><code>xfs.perdev.attr.*</code></td>
<td>Counts for the number of attribute get, set, remove and list operations over all XFS file systems.</td>
</tr>
<tr>
<td><code>xfs.perdev.quota.*</code></td>
<td>Metrics for quota operation over XFS file systems, these include counters for number of quota reclaims, quota cache misses, cache hits and quota data reclaims.</td>
</tr>
<tr>
<td><code>xfs.perdev.buffer.*</code></td>
<td>Range of metrics regarding XFS buffer objects. Counters include the number of requested buffer calls, successful buffer locks, waited buffer locks, miss_locks, miss_retries and buffer hits when looking up pages.</td>
</tr>
<tr>
<td><code>xfs.perdev.btree.*</code></td>
<td>Metrics regarding the operations of the XFS btree.</td>
</tr>
</tbody>
</table>
CHAPTER 6. SETTING UP GRAPHICAL REPRESENTATION OF PCP METRICS

Using a combination of `pcp`, `grafana`, `pcp redis`, `pcp bpftrace`, and `pcp vector` provides graphs, based on the live data or data collected by Performance Co-Pilot (PCP).

This section describes how to set up and access the graphical representation of PCP metrics.

6.1. SETTING UP PCP WITH PCP-ZEROCONF

This procedure describes how to set up PCP on a system with the `pcp-zeroconf` package. Once the `pcp-zeroconf` package is installed, the system records the default set of metrics into archived files.

**Procedure**

- Install the `pcp-zeroconf` package:
  
  ```sh
  # yum install pcp-zeroconf
  ```

**Verification steps**

- Ensure that the `pmlogger` service is active, and starts archiving the metrics:

  ```sh
  # pcp | grep pmlogger
  pmlogger: primary logger: /var/log/pcp/pmlogger/localhost.localdomain/20200401.00.12
  ```

**Additional resources**

- `pmlogger` man page
- Monitoring performance with Performance Co-Pilot

6.2. SETTING UP A GRAFANA-SERVER

Grafana generates graphs that are accessible from a browser. The `grafana-server` is a back-end server for the Grafana dashboard. It listens, by default, on all interfaces, and provides web services accessed through the web browser. The `grafana-pcp` plugin interacts with the `pmproxy` protocol in the backend.

This procedure describes how to set up a `grafana-server`.

**Prerequisites**

- PCP is configured. For more information, see Setting up PCP with pcp-zeroconf.

**Procedure**

1. Install the following packages:

   ```sh
   # yum install grafana grafana-pcp
   ```

2. Restart and enable the following service:
# systemctl restart grafana-server
# systemctl enable grafana-server

Verification steps

- Ensure that the **grafana-server** is listening and responding to requests:

  ```
  # ss -ntlp | grep 3000
  LISTEN 0 128 *:3000 *:* users:((“grafana-server”,pid=19522,fd=7))
  ```

- Ensure that the **grafana-pcp** plugin is installed:

  ```
  # grafana-cli plugins ls | grep performancecopilot-pcp-app
  performancecopilot-pcp-app @ 3.1.0
  ```

Additional resources

- **pmproxy**(1) and **grafana-server** man pages

### 6.3. ACCESSING THE GRAFANA WEB UI

This procedure describes how to access the Grafana web interface.

Using the Grafana web interface, you can:

- add PCP Redis, PCP bpftrace, and PCP Vector data sources
- create dashboard
- view an overview of any useful metrics
- create alerts in PCP Redis

Prerequisites

1. PCP is configured. For more information, see Setting up PCP with pcp-zeroconf.

2. The **grafana-server** is configured. For more information, see Setting up a grafana-server.

Procedure

1. On the client system, open a browser and access the **grafana-server** on port **3000**, using `http://192.0.2.0:3000` link.
   Replace `192.0.2.0` with your machine IP.

2. For the first login, enter `admin` in both the Email or username and Password field.
   Grafana prompts to set a New password to create a secured account. If you want to set it later, click Skip.

3. From the menu, hover over the Configuration icon and then click Plugins.
4. In the **Plugins** tab, type performance co-pilot in the **Search by name or type** text box and then click **Performance Co-Pilot (PCP)** plugin.

5. In the **Plugins / Performance Co-Pilot** pane, click **Enable**.

6. Click Grafana icon. The Grafana **Home** page is displayed.

**Figure 6.1. Home Dashboard**

![Home Dashboard](image)

**NOTE**

The top corner of the screen has a similar icon, but it controls the general **Dashboard settings**.

7. In the Grafana **Home** page, click **Add your first data source** to add PCP Redis, PCP bpftrace, and PCP Vector data sources. For more information on adding data source, see:

- To add pcp redis data source, view default dashboard, create a panel, and an alert rule, see [Creating panels and alert in PCP Redis data source](#).

- To add pcp bpftrace data source and view the default dashboard, see [Viewing the PCP bpftrace System Analysis dashboard](#).

- To add pcp vector data source, view the default dashboard, and to view the vector checklist, see [Viewing the PCP Vector Checklist](#).

8. Optional: From the menu, hover over the **admin profile** icon to change the **Preferences** including **Edit Profile, Change Password**, or to **Sign out**.

Additional resources

- **grafana-cli** and **grafana-server** man pages

**6.4. CONFIGURING PCP REDIS**
This section provides information for configuring PCP Redis data source.

Use the PCP Redis data source to:

- View data archives
- Query time series using pmseries language
- Analyze data across multiple hosts

**Prerequisites**

1. PCP is configured. For more information, see Setting up PCP with pcp-zeroconf.
2. The **grafana-server** is configured. For more information, see Setting up a grafana-server.

**Procedure**

1. Install the **redis** package:
   
   ```
   # yum install redis
   ```

2. Start and enable the following services:

   ```
   # systemctl start pmproxy redis
   # systemctl enable pmproxy redis
   ```

3. Mail transfer agent, for example, **sendmail** or **postfix** is installed and configured.

4. Ensure that the **allow_loading_unsigned_plugins** parameter is set to PCP Redis database in the **grafana.ini** file:

   ```
   # vi /etc/grafana/grafana.ini
   allow_loading_unsigned_plugins = pcp-redis-datasource
   ```

5. Restart the **grafana-server**:

   ```
   # systemctl restart grafana-server
   ```

**Verification steps**

- Ensure that the **pmproxy** and **redis** are working:

  ```
  # pmseries disk.dev.read
  2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df
  ```

  This command does not return any data if the **redis** package is not installed.

**Additional resources**

- **pmseries(1)** man page
6.5. CREATING PANELS AND ALERT IN PCP REDIS DATA SOURCE

After adding the PCP Redis data source, you can view the dashboard with an overview of useful metrics, add a query to visualize the load graph, and create alerts that help you to view the system issues after they occur.

Prerequisites

1. The PCP Redis is configured. For more information, see Configuring PCP Redis.
2. The grafana-server is accessible. For more information, see Accessing the Grafana web UI.

Procedure

1. Log into the Grafana web UI.
2. In the Grafana Home page, click Add your first data source
3. In the Add data source pane, type redis in the Filter by name or type text box and then click PCP Redis.
4. In the Data Sources / PCP Redis pane, perform the following:
   a. Add http://localhost:44322 in the URL field and then click Save & Test.
   b. Click Dashboards tab → Import → PCP Redis: Host Overview to see a dashboard with an overview of any useful metrics.

![Figure 6.2. PCP Redis: Host Overview](image)

5. Add a new panel:
   a. From the menu, hover over the icon to add a panel.
   b. In the Query tab, select the PCP Redis from the query list instead of the selected default option and in the text field of A, enter metric, for example, kernel.all.load to visualize the kernel load graph.
c. Optional: Add **Panel title** and **Description**, and update other options from the **Settings**.

d. Click **Save** to apply changes and save the dashboard. Add **Dashboard name**.

e. Click **Apply** to apply changes and go back to the dashboard.

**Figure 6.3. PCP Redis query panel**

6. Create an alert rule:

   a. In the **PCP Redis query panel** click **Alert** and then click **Create Alert**.

   b. Edit the **Name**, **Evaluate query**, and **For** fields from the **Rule**, and specify the **Conditions** for your alert.

   c. Click **Save** to apply changes and save the dashboard. Click **Apply** to apply changes and go back to the dashboard.

   **Figure 6.4. Creating alerts in the PCP Redis panel**

   d. Optional: In the same panel, scroll down and click **Delete** icon to delete the created rule.
Optional: From the menu, click **Alerting** icon to view the created alert rules with different alert statuses, to edit the alert rule, or to pause the existing rule from the **Alert Rules** tab. To add a notification channel for the created alert rule to receive an alert notification from Grafana, see Adding notification channels for alerts.

### 6.6. ADDING NOTIFICATION CHANNELS FOR ALERTS

By adding notification channels, you can receive an alert notification from Grafana whenever the alert rule conditions are met and the system needs further monitoring.

You can receive these alerts after selecting any one type from the supported list of notifiers, which includes DingDing, Discord, Email, Google Hangouts Chat, HipChat, Kafka REST Proxy, LINE, Microsoft Teams, OpsGenie, PagerDuty, Prometheus Alertmanager, Pushover, Sensu, Slack, Telegram, Threema Gateway, VictorOps, and webhook.

#### Prerequisites

1. The **grafana-server** is accessible. For more information, see Accessing the Grafana web UI.
2. An alert rule is created. For more information, see Creating panels and alert in PCP Redis data source.
3. Configure SMTP and add a valid sender's email address in the **grafana/grafana.ini** file:

```bash
# vi /etc/grafana/grafana.ini

[smtph]
enabled = true
from_address = abc@gmail.com
```

Replace *abc@gmail.com* by a valid email address.

#### Procedure

1. From the menu, hover over the **Alerting** icon → click Notification channels → Add channel.

2. In the Add notification channel details pane, perform the following:
   a. Enter your name in the Name text box
   b. Select the communication Type, for example, Email and enter the email address. You can add multiple email addresses using the ; separator.
   c. Optional: Configure Optional Email settings and Notification settings.

3. Click **Save**.

4. Select a notification channel in the alert rule:
   a. From the menu, hover over the **Alerting** icon and then click **Alert rules**.
b. From the Alert Rules tab, click the created alert rule.

c. On the Notifications tab, select your notification channel name from the Send to option, and then add an alert message.

d. Click Apply.

Additional resources

- Upstream Grafana documentation for alert notifications

6.7. SETTING UP AUTHENTICATION BETWEEN PCP COMPONENTS

You can setup authentication using the **scram-sha-256** authentication mechanism, which is supported by PCP through the Simple Authentication Security Layer (SASL) framework.

Procedure

1. Install the **sasl** framework for the **scram-sha-256** authentication mechanism:

   ```
   # yum install cyrus-sasl-scram cyrus-sasl-lib
   ```

2. Specify the supported authentication mechanism and the user database path in the **pmcd.conf** file:

   ```
   # vi /etc/sasl2/pmcd.conf
   
   mech_list: scram-sha-256
   
   sasldb_path: /etc/pcp/passwd.db
   ```

3. Create a new user:

   ```
   # useradd -r metrics
   ```

   Replace **metrics** by your user name.

4. Add the created user in the user database:

   ```
   # saslpasswd2 -a pmcd metrics
   
   Password:
   
   Again (for verification):
   ```

   To add the created user, you are required to enter the **metrics** account password.

5. Set the permissions of the user database:

   ```
   # chown root:pcp /etc/pcp/passwd.db
   # chmod 640 /etc/pcp/passwd.db
   ```

6. Restart the **pmcd** service:
# systemctl restart pmcd

Verification steps

- Verify the sasl configuration:

  ```
  # pminfo -f -h "pcp://127.0.0.1?username=metrics" disk.dev.read
  Password: disk.dev.read
  inst [0 or "sda"] value 19540
  ```

Additional resources

- `saslauthd(8)`, `pminfo(1)`, and `sha256` man pages

6.8. INSTALLING PCP BPFTRACE

Install the PCP bpftrace agent to introspect a system and to gather metrics from the kernel and user-space tracepoints.

The bpftrace agent uses bpftrace scripts to gather the metrics. The bpftrace scripts use the enhanced Berkeley Packet Filter (eBPF).

This procedure describes how to install a pcp bpftrace.

Prerequisites

1. PCP is configured. For more information, see Setting up PCP with pcp-zeroconf.

2. The grafana-server is configured. For more information, see Setting up a grafana-server.

3. The scram-sha-256 authentication mechanism is configured. For more information, see Setting up authentication between PCP components.

Procedure

1. Install the pcp-pmda-bpftrace package:

   ```
   # yum install pcp-pmda-bpftrace
   ```

2. Edit the bpftrace.conf file and add the user that you have created in the {setting-up-authentication-between-pcp-components}:

   ```
   # vi /var/lib/pcp/pmdas/bpftrace/bpftrace.conf
   
   [dynamic_scripts]
   enabled = true
   auth_enabled = true
   allowed_users = root,metrics
   
   Replace metrics by your user name.
   ```
3. Install `bpftrace` PMDA:

```
# cd /var/lib/pcp/pmdas/bpftrace/
# ./Install
Updating the Performance Metrics Name Space (PMNS) ...
Terminate PMDA if already installed ...
Updating the PMCD control file, and notifying PMCD ...
Check bpftrace metrics have appeared ... 7 metrics and 6 values
```

The `pmda-bpftrace` is now installed, and can only be used after authenticating your user. For more information, see Viewing the PCP bpftrace System Analysis dashboard.

Additional resources

- `pmdabpftrace(1)` and `bpftrace` man pages

### 6.9. VIEWING THE PCP BPFTRACE SYSTEM ANALYSIS DASHBOARD

Using the PCP bpftrace data source, you can access the live data from sources which are not available as normal data from the `pmlogger` or archives.

In the PCP bpftrace data source, you can view the dashboard with an overview of useful metrics.

Prerequisites

1. The PCP bpftrace is installed. For more information, see Installing PCP bpftrace.
2. The `grafana-server` is accessible. For more information, see Accessing the Grafana web UI.

Procedure

1. Log into the Grafana web UI.
2. In the Grafana Home page, click Add your first data source.
3. In the Add data source pane, type bpftrace in the Filter by name or type text box and then click PCP bpftrace.
4. In the Data Sources / PCP bpftrace pane, perform the following:
   b. Toggle the Basic Auth option and add the created user credentials in the User and Password field.
   c. Click Save & Test.
d. Click Dashboards tab → Import → PCP bpftrace: System Analysis to see a dashboard with an overview of any useful metrics.

6.10. INSTALLING PCP VECTOR

This procedure describes how to install a **pcp vector**.

**Prerequisites**

1. PCP is configured. For more information, see Setting up PCP with pcp-zeroconf.
2. The **grafana-server** is configured. For more information, see Setting up a grafana-server.

**Procedure**

1. Install the **pcp-pmda-bcc** package:
# yum install pcp-pmda-bcc

2. Install the bcc PMDA:

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

[Wed Apr 1 00:27:48] pmdabcc(22341) Info: Initializing, currently in 'notready' state.
[Wed Apr 1 00:27:48] pmdabcc(22341) Info: Enabled modules:
[Wed Apr 1 00:27:48] pmdabcc(22341) Info: ['biolatency', 'sysfork', ...
[...] Updating the Performance Metrics Name Space (PMNS) ... Terminating PMDA if already installed ... Updating the PMCD control file, and notifying PMCD ... Check bcc metrics have appeared ... 1 warnings, 1 metrics and 0 values
```

Additional resources

- `pmdabcc(1)` man page

6.11. VIEWING THE PCP VECTOR CHECKLIST

The PCP Vector data source displays live metrics and uses the `pcp` metrics. It analyzes data for individual hosts.

After adding the PCP Vector data source, you can view the dashboard with an overview of useful metrics and view the related troubleshooting or reference links in the checklist.

Prerequisites

1. The PCP Vector is installed. For more information, see Installing PCP Vector.

2. The grafana-server is accessible. For more information, see Accessing the Grafana web UI.

Procedure

1. Log into the Grafana web UI.

2. In the Grafana Home page, click Add your first data source

3. In the Add data source pane, type vector in the Filter by name or type text box and then click PCP Vector.

4. In the Data Sources / PCP Vector pane, perform the following:
   a. Add `http://localhost:44322` in the URL field and then click Save & Test.
   b. Click Dashboards tab → Import → PCP Vector: Host Overview to see a dashboard with an overview of any useful metrics.
6.12. TROUBLESHOOTING GRAFANA ISSUES

This section describes how to troubleshoot Grafana issues, such as, Grafana does not display any data, the dashboard is black, or similar issues.

**Procedure**

- Verify that the **pmlogger** service is up and running by executing the following command:

  ```bash
  $ systemctl status pmlogger
  ```
• Verify if files were created or modified to the disk by executing the following command:

```bash
$ ls /var/log/pcp/pmlogger/$(hostname)/ -rlt
```

total 4024

-rw-r--r--. 1 pcp pcp 45996 Oct 13 2019 20191013.20.07.meta.xz
-rw-r--r--. 1 pcp pcp 412 Oct 13 2019 20191013.20.07.index
-rw-r--r--. 1 pcp pcp 32188 Oct 13 2019 20191013.20.07.0.xz
-rw-r--r--. 1 pcp pcp 44756 Oct 13 2019 20191013.20.30-00.meta.xz

[...]

• Verify that the `pmproxy` service is running by executing the following command:

```bash
$ systemctl status pmproxy
```

• Verify that `pmproxy` is running, time series support is enabled, and a connection to Redis is established by viewing the `/var/log/pcp/pmproxy/pmproxy.log` file and ensure that it contains the following text:

```
pmproxy(1716) Info: Redis slots, command keys, schema version setup
```

Here, 1716 is the PID of pmproxy, which will be different for every invocation of `pmproxy`.

• Verify if the Redis database contains any keys by executing the following command:

```bash
$ redis-cli dbsize
```

(integer) 34837

• Verify if any PCP metrics are in the Redis database and `pmproxy` is able to access them by executing the following commands:

```bash
$ pmseries disk.dev.read
2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df

$ pmseries "disk.dev.read[count:10]"
2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df
  70e83e88d4e1857a3a31605c6d133375f2dd17c
  [Mon Jul 26 12:21:00.087401000 2021] 117758
  70e83e88d4e1857a3a31605c6d133375f2dd17c
  70e83e88d4e1857a3a31605c6d133375f2dd17c
  [...

$ redis-cli --scan --pattern "+\$(pmseries 'disk.dev.read')"

pcp:metric.name:series:2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df
pcp:values:series:2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df
pcp:desc:series:2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df
pcp:labelflags:series:2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df
pcp:instance.value:series:2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df
pcp:labelflags:series:2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df
```

• Verify if there are any errors in the Grafana logs by executing the following command:
$ journalctl -e -u grafana-server
-- Logs begin at Mon 2021-07-26 11:55:10 IST, end at Mon 2021-07-26 12:30:15 IST. --
Jul 26 11:55:17 localhost.localdomain systemd[1]: Starting Grafana instance...
lvl=info msg="Starting Grafana" logger=server version=7.3.6
lvl=info msg="Config loaded from" logger=settings file=/usr/s
lvl=info msg="Config loaded from" logger=settings file=/etc/g
[...]

Red Hat Enterprise Linux 9.0 Beta Monitoring and managing system status and performance