Monitoring of Virtual Environments

Monitoring of Virtual Environments

Discovery PFMS 2.0 Applications is used to monitor
Amazon EC2®, DB2®, SAP® and VMware®.

RHEV

Red Hat® Enterprise Virtualization (RHEV) is one of the most widely used technologies by companies that have Red Hat operating system as a base in their Data Center. Pandora FMS Enterprise offers the possibility of monitoring virtual architectures based on RHEV by means of the RHEV Monitoring Plugin.

Architecture to monitor

- Data Centers.
- Host Clusters.
- Storage Domains.
- Networks.
- Hosts.
- Virtual Machines.

Pandora FMS uses the official API provided by the RHEV virtualization system.

Monitoring with RHEV Monitoring Plugin

For the monitoring of the operating system installed in the virtual machines it is recommended to use a Pandora FMS Agent instead of the RHEV API.

Virtual environment monitoring RHEV is based on two components:

1. An Agent plugin that performs the auto-discovery and data collection tasks. The Agent plugin is in charge of sending the information to Pandora FMS - A recon script that updates several values for the discovered entities, necessary for the correct functioning of the plugin extensions - RHEV Viewer and RHEV Manager: They are extensions that provide the operation of turning off/on virtual machines, all from the Pandora FMS Web Console.

To be able to use the discovery script it is necessary to have the Discovery server activated. In order for certain API variables to reflect the actual value of the associated virtual machine you need to install the RHEV Agent; you

can find all about it in the documentation of your RHEV
version.

Plugin inner workings

The RHEV Monitoring Plugin extracts the information via the web API served by the RHEV virtualization environment.

If you only need the monitoring information all you have to configure is the Agent plugin that will perform this task. The plugin configuration allows you to choose which elements are going to be monitored and the configuration of its Modules. Once the XMLs are created, the Agent plugin sends the files, either using Tentacle or copies them to a local directory, depending on the chosen transfer method.

If you are also going to use the RHEV Viewer and RHEV Manager extensions you will need to use the recognition script. The recognition script is in charge of updating some variables for each one of the Agents detected in Pandora FMS according to the values configured in RHEV. These variables are necessary to be able to visualize the entities correctly in the RHEV Viewer extension and to manage properly the virtual machines with the RHEV Manager extension.

Installation Prerequisites

The Agent plugin requires the following software:

- curl
- perl-XML-Simple
- Software Agent and Pandora FMS `tentacle_client`.

Red Hat

On Red Hat® based systems you can install the dependencies with the command:

```
yum install perl-XML-Simple curl
```

SLES

On SUSE-based systems you can install the dependencies with the command:

```
zypper install perl-XML-Simple curl
```

Debian/Ubuntu

On Debian/Ubuntu based systems you can install the dependencies with the command:

```
apt-get install libxml-simple-perl curl
```

RHEV certificate download

Before using the plugin it will be necessary to download the certificate that allows HTTPS connection to the RHEV API. To do this, run the following command:

```
curl -o rhevm.cer http://[RHEVM-HOST]:8080/ca.crt
```

Where `[rhevm-host]` is the name of the server serving the RHEV API. A concrete example could be:

```
curl -o rhevm.cer http://rhevm.server:8080/ca.crt
```

Once the certificate is downloaded you can check that the API connection is successful with the following command using > line connectors.

```
curl -X GET \ -H "Accept: application/xml" \ -u [USER:PASS] \ --cacert [CERT]
https://[RHEVM-HOST]:8443/api
```

With the following values:

- USER: user @domain to connect to the API.
- PASS: password of the user with which you will connect to the API.
- CERT: path to the certificate downloaded in the previous step.
- RHEVM-HOST: address of the host serving the API.

Example with concrete data of the command:

```
curl -X GET \ -H "Accept: application/xml" \ -u [user@testdomain:12345] \ --
cacert /home/user/ca.crt https://rhevm.server:8443/api
```

If the command execution is positive, it will return an output in XML format with general information about the RHEV API.

Preliminary considerations about RHEV configuration

In the RHEV virtualization environment it is possible for several entities to have the same name. This is a problem, because in Pandora FMS these entities will be transformed into Agents in which duplicity in the names is not allowed. In addition, it will also generate problems when parsing the result returned by the API in XML format, showing an error similar to the following:

```
Warning: <data_center> element has non-unique value in 'name' key attribute:
Default at ./plugin-rhev.pl line 199
```

To solve the problem the only thing necessary is to follow a naming nomenclature for the entities in the RHEV virtualization environment in which the names are not repeated.

Installing the Agent plugin

To install the Agent plugin you just have to copy the script `rhev-plugin.pl` and the configuration file `rhev-plugin.conf` to a directory on the machine where the Pandora FMS Agent is installed. will run the plugin. The plugin can be run on an Agent installed on the same machine as the Pandora FMS server or on a different machine.

To run the plugin you must add the following line to the Agent configuration file (by default `/etc/pandora/pandora_agent.conf`):

```
module_plugin /root/rhev-plugin.pl /root/rhev-plugin.conf
```

By adding this line the Agent plugin will perform its functions on each execution.

Monitoring the RHEV virtual architecture

To see the result of running the Agent plugin go to Operation → Monitoring → Views → Agent Detail. By clicking on the name of an Agent you will be able to see the Monitoring Modules created by the plugin, as well as other relative data.

The plugin creates an Agent in Pandora FMS for each of the entities detected in the discovery of the RHEV architecture. For each type of entity, a series of specific Modules are automatically created, monitoring the important information of each of them.

If the selected Agent corresponded to a Host instead of a Virtual Machine, the monitoring modules would be different.

The RHEV plugin also monitors events that occur within the virtual architecture. The plugin will create a Module for each monitored event within each affected entity. The data for Modules created from events is event data: event time, event description.

In addition to the Agents and Modules related to the RHEV architecture itself, a Module called, by default RHEV Plugin, is generated in the Agent that executes the plugin.

RHEV Virtual Architecture Agent Modules

Data Center

- Status: Data Center status.

Storage Domain

- Available Space: Available space in the Storage Domain.
- Committed Space: Committed space in the Storage Domain.
- Used Space: Used space in the Storage Domain.
- Percent Free Space: Percentage of free space in the Storage Domain.

Network

- Status: Status of the virtual network.
- STP Status: Status of the Spanning Tree Protocol functionality.

Cluster

- Overcommit Percent: Cluster overcommit percentage.
- Transparent HugePages: Status of the Transparent HugePages functionality.
- High threshold: Upper limit in planning policies.
- Low threshold: Lower limit in planning policies.
- Threshold duration: Duration of the limits in planning policies.

Host

- Status: Host status.
- Buffers size: Size of the buffers.
- Cache size: Cache size.
- Cached swap: Amount of Swap cache memory (in bytes).
- Free memory: Amount of free memory (in bytes).
- Percent free memory: Percentage of free memory.
- Swap cached percent: Swap cache memory percentage.
- Swap free: Amount of free Swap memory (in bytes).
- Swap free percent: Percentage of free Swap memory.
- Total Memory: Total amount of Host memory (in bytes).
- Total Swap: Total amount of Swap memory (in bytes).
- Used memory: Total amount of memory used (in bytes).
- Used Swap: Total amount of Swap memory used (in bytes).
- Nic [x] TX and Nic [x] RX: Network interface transfer rate [x] (in bytes/second). One is generated for each detected network interface.
- Nic [x] errors TX and Nic [x] errors RX: Number of network interface [x] transmission errors. One is generated for each detected network interface.
- User CPU: Percentage of CPU used by the user.
- System CPU: Percentage of CPU used byr the system.

- CPU Idle: Percentage of idle CPU.
- CPU Load: Average CPU load for the last 5 minutes.
- KSM CPU: Percentage of CPU used by KSM.
- Active VM: Number of active virtual machines on the Host.
- Migrating VM: Number of virtual machines being migrated on the Host.
- Total VM: Total number of virtual machines on the Host.
- Fence Status: Host fencing status.

Virtual Machine

- Status: State of the virtual machine.
- Disk [x] read and Disk [x] write: Read and write rate of disk x (bytes/second). One is generated for each disk (storage) detected.
- Disk [x] size: Size of disk x (in bytes). One is generated for each detected disk.
- Disk [x] status: Status of disk x. One is generated for each detected disk.
- Nic [x] TX and Nic [x] RX: Transfer and reception rate for network interface [x] (in bytes/second). One is generated for each detected network interface.
- Nic [x] errors TX and Nic [x] errors RX: Number of transmission and reception errors for the network interface [x]. One is generated for each detected network interface.
- Installed memory: Amount of installed memory (in bytes).
- Percent free memory: Percentage of free memory.
- Used memory: Amount of memory used (in bytes).
- Stateless: State of the Stateless functionality.
- HA Status: Status of the HA functionality.
- Total CPU: Total percentage of CPU used by the virtual machine.
- Hypervisor CPU: Percentage of Hypervisor CPU used by the virtual machine.
- Guest CPU: Percentage of Host CPU used by the virtual machine.

Events

- Event [x]: Description of event x that occurred in the system. One will be created for each event detected in the affected Agents.

RHEV architecture management and visualization

Recognition tasks

There is the possibility of creating custom recognition tasks thanks to the Discovery server.

Installing RHEV View and RHEV Manager extensions

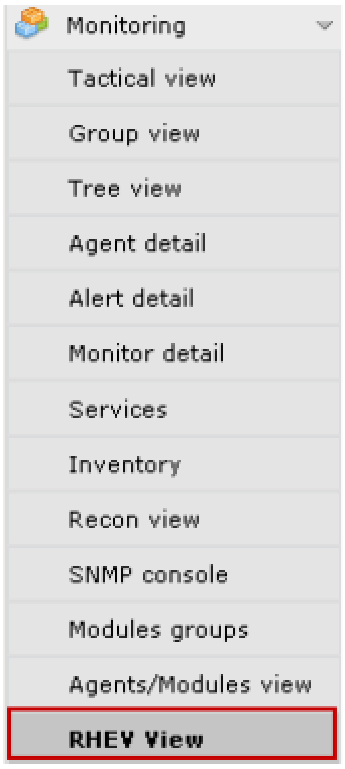
To install the extensions you simply have to copy the contents of the `extensions` folder, which you will find when unzipping the plugin, into the corresponding `extensions` folder in the Enterprise part of the Pandora FMS Console. The command to execute is the following:

```
cp -R extensions/* <pandora_console_dir>/enterprise/extensions/
```

From that moment on, RHEV monitoring extensions will be available.

Using the RHEV View extension

To use the RHEV View extension you just have to click on the RHEV View option within the Monitoring submenu.



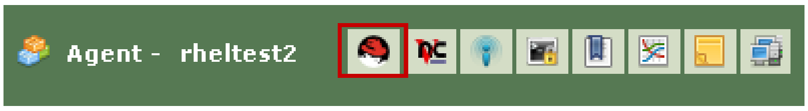
The extension will display a map with all the RHEV architecture components discovered by the plugin.

Using the RHEV Manager extension

The RHEV Manager extension is available in the operation view of Pandora FMS agents that correspond to virtual machines within the RHEV virtualization architecture.

This extension uses the curl command, so it will need to be installed and accessible to the web server that supports the Pandora FMS Console.

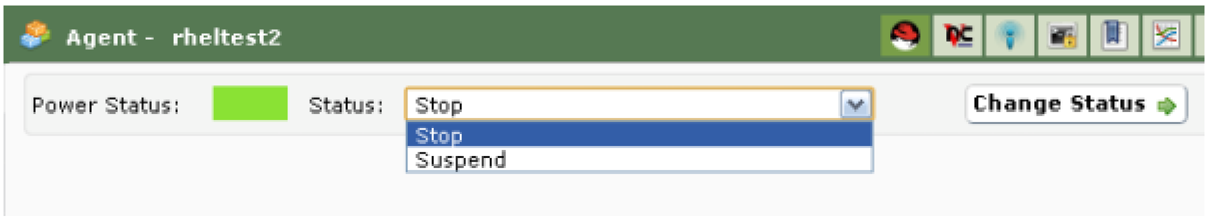
To access the extension, click on the button with the Red Hat logo that you will find along with the other agent tabs.



The extension allows you to manage virtual machines (power on, shut down and suspend) without opening the RHEV management console. The extension shows the current state of the virtual machine with a color code:

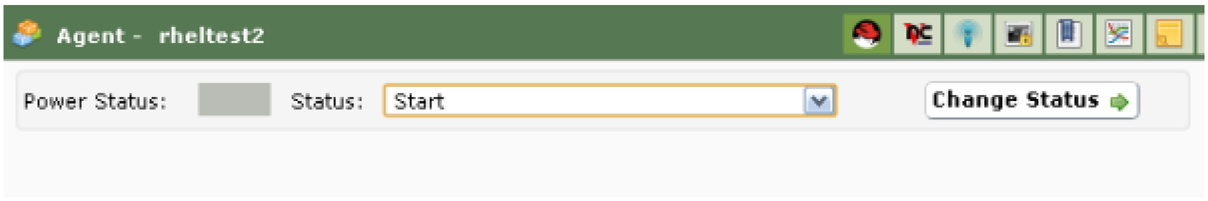
- Green = On.
- Orange = Suspended.
- Gray = Stopped.

With a combo with the available states to the states that the virtual machine can be taken by pressing the Change Status button.



If you choose the Stop state to stop the virtual machine, the extension will connect to the RHEV API and send the command. The result will be the change of state in the virtual machine and the combo options.

The transition between some states is not automatic, such as from Stop state to Start. In this case, the extension will show the state of the virtual machine as it changes in the virtualization architecture.



Agent Plugin Configuration

The configuration of the Agent plugin is done using a configuration file whose default name is `rhev-plugin.conf`.

By default, the Agent plugin selects all entities and creates all corresponding Modules with default values for the name and description. All these aspects, as well as general plugin variables, can be configured through the configuration file.

Configuration file

The configuration file has two distinct areas: the g variableslobal and monitoring configuration.

The global variables section starts with the Configuration token and contains the plugin's configuration information. The parameters allowed in this section are:

- module_name: Name of the Agent Module with the status of the plugin execution.
- server: Name of the host that serves the RHEV API.
- user: User in user @domain format to connect to the API.
- pass : Password to connect to the API.
- cert: Certificate path to connect to the API.
- temporary: Temporary directory.
- logfile: Log file.
- transfer_mode : Transfer mode. It can take the values: `local` or `tentacle`.
- tentacle_ip: IP address of the Tentacle server to which to send the information. Typically it will be located on the same machine as the Pandora FMS server. This option is only used if `transfer_mode` has the value `tentacle`.
- tentacle_port: Tentacle server port. This option is only used if `transfer_mode` has the value `tentacle`.
- tentacle_opts: Data sending options for Tentacle. This option is only used if `transfer_mode` has the value `tentacle`.

The monitoring configuration section is divided into several subsections. The first subsection has Reject as a token and serves to list the entities in the virtualization environment that will be discarded from monitoring. To discard an entity it will be necessary to put its name in this list. For example:

```
#Dismissed entities
Reject
mv1
mv_Windows10
mv_WebServer1
...
```

It is possible to discard all entities of the same type, for example all hosts, all virtual machines, etc. The tokens for each entity are: all_dc (Data Center), all_host (Hosts), all_network (Networks), all_storage (Storage Domain), all_cluster (Cluster), all_vm (Virtual Machines). Example of use of these tokens:

```
#Dismissed entities
Reject
all_dc
all_host
all_network
all_storage
all_cluster
all_vm
```

The second section has Rename as a token and is used to change the names of the entities monitored through the plugin. This functionality is very useful if you want to combine the monitoring of software agents with data extracted from the API in the Pandora FMS Agent itself. The configuration of this section is done by putting the old name first and then the new one separated by a space; For example:

```
  #Rename entities
  Rename
  mv_WebServer1 WebServer1
  mv_Windows10 Windows10 Test
  ...
```

The following subsections correspond to the monitoring configuration for each entity. Each entity has its own token, being the following: DataCenter, StorageDomain, Network, Cluster, Host and VM. For each of these entities it is possible to define the Modules that will be disabled or define custom values for the name, description and the maximum and minimum ranges for the `Warning` and `Critical` states. An example would be the following:

```
#VMModules
V.M.
disabled status
errors_total_tx name = Net TX Errors [%s]; desc = Total Network TX Errors;
limits = 60 70 71 100
memory_used name = Memory in use; desc = Memory used by the virtual machine;
limits = 256 1024 1025 2048
...
```

Each configuration line of the monitoring modules corresponds to two available options:

- <modulo> disabled: The Module will NOT be created.
- <module> name = <name>; desc = <description>; limits = <min_warning> <max_warning> <min_critical> <max_critical» The Module will be created with the name and description provided and the thresholds for the maximums and minimums of the `Warning` and `Critical` values will also be defined '.

It is very important to take into account the structure of the configuration file lines and above all to see that the character `;` is attached to the name and description of the module. These two lines are not equivalent (see the spaces before the `;` character):

```
errors_total_tx name = Net TX Errors [%s]; desc = Total Network TX Errors;
limits = 60 70 71 100 #Correct
errors_total_tx name = Net TX Errors [%s] ; desc = Total Network TX Errors ;
limits = 60 70 71 100 #Incorrect
```

Modules are referenced by their short name, an equivalent name that is easier to type on the command line. The correspondence table between short and expanded names is in the next section.

Configuration example for virtual machines, VM section:

To monitor virtual machines, a series of Modules, enabled or not, have been defined in the VM section of the configuration file. More specifically: Module status has been disabled and custom values have been defined for Modules errors_total_tx and memory_useds. The other Modules that do not appear in the list will be created with the default values. With this configuration, Module memory_used will take the following values:

- Name: Memory in use.
- Description: Memory used by the virtual machine.
- Min Warning: 256.
- Max Warning: 1024.
- Min Critical: 1025.
- Max Critical: 2048.

Modules are generated dynamically; for example, two related to disks or interfaces of which one is created for each element detected. They have a special syntax for the Module name, which is as follows:

```
errors_total_tx name = Net TX Errors [%s]; desc = Total Network TX Errors;
limits = 60 70 71 100
```

In these cases, since the name has a dynamic part, what is allowed is to use the %s macro that will be replaced by the plugin with the variable part of the Module name.

For example, Module errors_total_tx has the default name:

```
Nic [nic1] errors TX
```

It will be called

```
TX Net Errors [nic1]
```

nic1 being the dynamic part of the module name.

All errors related to the configuration file are presented in the log defined in the configuration file and are also sent as an asynchronous Module to Pandora FMS that will be reflected as a Module within the Agent that executes the plugin.

In addition to the sections specific to each element of the architecture, the configuration file has a common section for Events. This section is defined with the token EventCodes and it will list the codes of the events to be monitored; For example:

```
EventCodes
30
920
980
509
956
```

If you do not define this section, event monitoring will not be performed.

Split the monitoring load between several Software Agents

Using the Agent plugin configuration file it is possible to divide the monitoring load of the RHEV virtualization infrastructure.

To do this, the entities to be monitored will be distributed among the different Agents. Suppose you have the following architecture:

```
DC1
 |
 |- Cluster 1.1
       |- c1.1mv1
       |- c1.1mv2
       |- c1.1mv3

 |- Cluster 1.2
       |- c1.2mv1
       |- c1.2mv2
       |- c1.2mv3

DC2
 |
 |- Cluster 2.1
       |- c2.1mv1
       |- c2.1mv2
       |- c2.1mv3

 |- Cluster 2.2
       |- c2.2mv1
       |- c2.2mv2
       |- c2.2mv3
```

One way to divide the load would be to assign a Datacenter to each of the Software Agents; For this we would use the functionality to discard entities to be monitored (Reject token).

The first Software Agent monitors Datacenter DC1 and discards the entities of DC2.

```
Reject
```

```
DC2
Cluster 2.1
Cluster 2.2
c2.1mv1
c2.1mv2
c2.1mv3
c2.2mv1
c2.2mv2
c2.2mv3
```

The second Software Agent monitors Datacenter DC2 and discards the entities of DC1.

```
Reject
DC1
Cluster 1.1
Cluster 1.2
c1.1mv1
c1.1mv2
c1.1mv3
c1.2mv1
c1.2mv2
c1.2mv3
```

We could also split the load based on clusters, for example. For each cluster of the two Datacenters, an agent from the first four will be assigned.

Software Agent 1, monitor `Cluster 1.1` and discard the other entities.

```
Reject
DC1
Cluster 1.2
c1.2mv1
c1.2mv2
c1.2mv3
DC2
Cluster 2.1
Cluster 2.2
c2.1mv1
c2.1mv2
c2.1mv3
c2.2mv1
c2.2mv2
c2.2mv3
```

Software Agent 2, monitor `Cluster 1.2` and discard the other entities.

```
Reject
DC1
Cluster 1.1
c1.1mv1
```

```
  c1.1mv2
  c1.1mv3
  DC2
  Cluster 2.1
  Cluster 2.2
  c2.1mv1
  c2.1mv2
  c2.1mv3
  c2.2mv1
  c2.2mv2
  c2.2mv3
```

Software Agent 3, monitor `Cluster 2.1` and discard the other entities.

```
  Reject
  DC1
  Cluster 1.1
  Cluster 1.2
  c1.1mv1
  c1.1mv2
  c1.1mv3
  c1.2mv1
  c1.2mv2
  c1.2mv3
  DC2
  Cluster 2.2
  c2.2mv1
  c2.2mv2
  c2.2mv3
```

Software Agent 4, monitor `Cluster 2.2` and discard the other entities.

```
  Reject
  DC1
  Cluster 1.1
  Cluster 1.2
  c1.1mv1
  c1.1mv2
  c1.1mv3
  c1.2mv1
  c1.2mv2
  c1.2mv3
  DC2
  Cluster 2.1
  c2.1mv1
  c2.1mv2
  c2.1mv3
```

The configuration of discarded entities is totally flexible and the load could be divided by assigning several entities to each Software Agent.

Nutanix

How the Nutanix plugin works

The Nutanix® plugin is a program written in Perl, which will connect to the Nutanix PRISM® REST API, retrieving the necessary metrics to monitor the following elements:

- Nutanix® clusters.
- Storage devices.
- Containers.
- Virtual machines.
- Hosts.
- Status of replication processes.

Nutanix Plugin Requirements

In order to retrieve information from the REST API, you need:

- The IP address/FQDN of the portal.
- A user with read permissions on the API.
- The password of said user.

As for the communitynication of the monitoring results to your Pandora FMS requires:

- The mode of information transfer, whether local or via Tentacle.
  - If local, the address of the directory where the XML files with the results are to be delivered, as well as writing permissions in said directory.
  - In case of communication via Tentacle, it will be necessary to be able to connect against the IP address or FQDN of the Pandora FMS server, the port used by your Tentacle installation, the location of the Tentacle client as well as any extraordinary option that you have defined.

Nutanix plugin installation

Download the files required by the plugin from the Modules library. Transfer the files to the remote computer where you want to monitor your Nutanix® infrastructure and extract the plugin files:

```
tar xvzf pandora_nutanix.tar.gz
```

Nutanix Plugin Configuration

The following fields are declared:

Nutanix API configuration

- nx_fqdn: Address of the main Prism server.
- nx_port: Port on which the REST API is published (default 9440).
- nx_user: User with reading privileges over the REST API.
- nx_pass: Password of said user.
- use_https: Use https (1) or not (0)
- nx_rest_version: Rest API version (default 'v1').

Nutanix agent configuration

- agent_interval: Interval of the Agents generated by the plugin (default 300).
- agent_group: Group to which the generated Agents will belong (if 'autocreate_group' is commented out in your PandoraServer configuration), by default 'Nutanix'.
- module_interval: Interval of the Modules of the generated agents (multiplication factor, default 1).
- module_tags: Tags associated with the new modules of the generated agents.
- module_group: Group to which the new Modules will belong.

Configuration of communication to the Pandora FMS server

- mode: Data transfer mode, “local” or “tentacle”.
- tentacle_ip: IP address of the Pandora FMS server, only applicable in Tentacle mode.
- tentacle_port: Port on which the Tentacle service is listening.
- tentacle_opts: Any extra options you have configured in your Tentacle service.
- tentacle_client: Full path to your Tentacle client.
- temp: Temporary working directory.
- local_folder: Delivery path for “local” data transfer mode.

Filters

- cluster_monitoring: Enable (1) or not (0) cluster monitoring.
- storage_monitoring: Enable (1) or not (0) storage device monitoring.
- container_monitoring: Enable (1) or not (0) monitoring of storage containers.
- vm_monitoring: Enable (1) or not (0) virtual machine monitoring.
- host_monitoring: Enable (1) or not (0) monitoring of virtual machine servers (Nutanix nodes).
- pd_monitoring: Enable (1) or not (0) protection domain monitoring.

Customizations

- cluster_agent_header: Header for the name of the cluster device agent.
- storage_agent_header: Header for the name of the Storage Device type Device Agent.
- host_agent_header: Header for the name of the virtual machine server type Device Agent (Nutanix nodes).
- container_agent_header: Header for the name of the storage container type Device Agent.
- vm_agent_header: Header for the name of the virtual machine type Device Agent.
- pd_agent_header: Header for the name of the Protection Domain Type Device Agent

Module generation rules

- vm_stat: Rule for adding Modules for monitoring virtual machines, by default `hypervisor_cpu_usage_ppm|hypervisor_memory_usage_ppm|.*avg.*`, this indicates the extraordinary Modules that will be generated, when the metric name matches the regular expressions indicated in this field. Add the value `.*` to monitor all available metrics.
- host_stat: Rule for adding Modules for monitoring virtual machine servers (Nutanix nodes), by default

  `hypervisor_cpu_usage_ppm|hypervisor_memory_usage_ppm|.*avg.*`, this indicates the extraordinary Modules that will be generated, when the metric name matches the regular expressions indicated in this field. Add the value `.*` to monitor all available metrics.
- pd_stat: Rule for adding Modules for monitoring protection domains, by default `replication_transmitted_bandwidth_kBps|replication_total_transmitted_bytes`, this indicates the extraord Modulesinaries that will be generated, when the metric name matches the regular expressions indicated in this field. Add the value `.*` to monitor all available metrics.

Renaming entities

- RENAME aaa TO bbb: Rule for renaming entities, you can define as many directives as elements you need to rename.

Exclusion of entities

- REJECT aaa: Rule for excluding entity monitoring, you can define as many policies as elements you need to exclude.

Running the Nutanix plugin

It is recommended to run the plugin remotely from a computer with access to both Pandora Server and your Nutanix® infrastructure to be monitored.

Manual execution:

```
./pandora_nutanix-linux-x64 pandora_nutanix.conf
```

You can automate the execution of the plugin in the system cron by adding the following line to `/etc/crontab>`

```
/5 * * * * root /path/to/plugin/pandora_nutanix-linux-x64
/path/to/plugin/pandora_nutanix.conf
```

XenServer

How the XenServer plugin works

The Pandora FMS plugin for monitoring Xen environments is written in Python. Use XenAPI to retrieve all necessary information. Allows monitoring of the following types of elements:

- Virtualized systems in Xen.
- Storage resources.
- Own XenServer 6.5 and 7.2 (host).

XenServer Plugin Requirements

It is essential that the system that runs the plugin has the following requirements:

- Python installed
- Installed Python libraries:
  - XenAPI
  - xmltodict
- Access to your XenServer API (web, enable traffic from the computer running the plugin to port 443 or 80 of the XenServer).
- It is recommended that virtual machines have Xen Server Tools installed, as the information available otherwise is quite limited.

Plugin installation

Download your copy of the Pandora FMS plugin for the contents of the file in a non-volatile directory from where you can execute it, either using the Pandora FMS Agent or the system cron.

Plugin Settings

Configuration available for the Pandora FMS plugin for Xen:

Configuration block [CONF]

- xen_server_ip: IP address/FQDN of the Xen server.
- user: User with query permissions on the Xen API.
- password: User password
- temporary: Temporary working directory

Configuration Block [PANDORA]

- tentacle_client: Location of the Tentacle client binary.
- tentacle_ip: IP address where the Tentacle service is listening.
- tentacle_port: Port where the Tentacle service is listening.
- logfile: Full path to the log file.
- interval: Interval of the generated agents.
- group: Group assigned to the generated agents.

Configuration block [TUNNING]

- time_adjustment: Parameter that allows the adjustment of possible time differences between the computer running the plugin and the Xen server. (default=10, measured in seconds)
- scan_vm_ip: Parameter that allows you to define whether the plugin will try to obtain the IPs of the VMs from the Xen server. Only the IPs of those VMs with XenTools installed can be taken. It can be enabled (scan_vm_ip=true) or disabled (scan_vm_ip=false). If not specified, it is considered enabled.

[RENAME] configuration block

- xen_element_name = pandora_agent_name: In this block you can define as many entries as you want with this format. Allows you to change the names of the Xen Server elements to different ones to be used as agent names in Pandora FMS. VMs, SRs and the Xen Server itself can be renamed.

Running the plugin

You can schedule the plugin to run from any Pandora FMS agent, adding the following to its configuration:

```
module_plugin python "<path>\xen-plugin.py" "<path>\xen-plugin.conf"
```

To schedule it through the system cron you can add the following line to /etc/crontab:

```
/5 * * * * root python "<path>\xen-plugin.py" "<path>\xen-plugin.conf">
/dev/null 2>&1
```

If you run the plugin manually the output should look like this:

```
python "<path>\xen-plugin.py" "<path>\xen-plugin.conf"
<module>
<name><![CDATA[XenServer Plugin]]></name>
<type><![CDATA[async_string]]></type>
<description><![CDATA[Result of XenServer Plugin execution]]></description>
<data><![CDATA[OK]]></data>
</module>
```

OpenNebula

How the OpenNebula plugin works

The Pandora FMS plugin for monitoring OpenNebula environments is written in Perl. Runs locally on the serveridor OpenNebula and will retrieve all the necessary information using OpenNebula's own management commands. Allows monitoring of the following types of elements:

- Clusters.
- Hosts.
- Virtual machines.
- Storage resources.

OpenNebula Plugin Requirements

It is essential that the system that runs the plugin has the following requirements:

- Perl available on your computer

- User with privileges to execute the following commands:
    - `onehost`.
    - `onecluster`.
    - `onedatastore`.

The operation of the plugin has been satisfactorily tested
on OpenNebula 5.X.X systems.

Plugin installation

Download your copy of the Pandora FMS plugin for OpenNebula from the module library. You must extract the contents of the file to a non-volatile directory where you can run it, either using the Pandora FMS agent or the system cron.

```
unzip pandora_OpenNebula.zip
```

Plugin Settings

Configuration available for Pandora FMS plugin for OpenNebula.

Communication configuration to the Pandora FMS server

- mode: Data transfer mode, “local” or “tentacle”.
- tentacle_ip: IP address of the Pandora FMS server, only applicable in tentacle mode.
- tentacle_port: Port on which the Tentacle service is listening.
- tentacle_opts: Any extra options you have configured in your Tentacle service.
- tentacle_client: Full path to your Tentacle client.
- temp: Temporary working directory.
- local_folder: Delivery path for “local” data transfer mode.

Agent Configuration

- agent_interval: Agent interval, default 300.
- agent_group: Agent group, by default OpenNebula.

Module Customization

- MODULE_GROUP: Group of modules, by default OpenNebula.
- MODULE_INTERVAL: Module interval (multiplier), default 1.
- MODULE_TAGS: Tags for the Modules.

Name customization

- cluster_agent_header: Header for the name of the cluster device agent.
- host_agent_header: Header for the name of the Virtual Machine Server type Device Agent.
- storage_agent_header: Header for the name of the Storage Device type Device Agent.
- vm_agent_header: Header for the name of the virtual machine type Device Agent.

Filters

- cluster_monitoring: Enable (1) or not (0) cluster monitoring.
- host_monitoring: Enable (1) or not (0) monitoring of virtual machine servers.
- storage_monitoring: Enable (1) or not (0) storage device monitoring.
- vm_monitoring: Enable (1) or not (0) virtual machine monitoring.

Renaming entities

RENAME aaa TO bbb: Rule for renaming entities, you can define as many directives as elements you need to rename.

Exclusion of entities

REJECT aaa

Rule for excluding entity monitoring, you can define as many policies as elements you need to exclude.

Running the plugin

To schedule it through the system cron you can add the following line to /etc/crontab:

```
/5 * * * * root "<path>/pandora_opennebula" "<path>/pandora_opennebula.conf">
/dev/null 2>&1
```

If you run the plugin manually the output should look like this:

```
[root@valhalla ~]# ./pandora_opennebula pandora_opennebula.conf
[root@valhalla ~]# echo $?
0
```

IBM HMC

This plugin allows you to monitor IBM AIX virtualization machines through the HMC hardware management console. This plugin will collect information from all logical partitions created in an AIX environment managed by an HMC system, creating an Agent for each managed server, each logical partition and each virtual IO server.

To collect information via SSH, the plugin can use three working modes:

1. Based on expect using the `ssh_launcher.sh script`.
2. Based on `Net::SSH::Perl`
3. Based on `Net::SSH::Expect`

To complement the information captured, queries will also be made against the REST API, by default in:

```
https://fqdn:12443/rest/api/{root_element}
```

Requirements

The necessary parameters that the area that requires monitoring services must provide are:

- Username to authenticate to the HMC system (read-only).
- The user must have permission to connect to the REST API and to log in to the HMC shell and execute the following commandsandos (at least):
  - lssyscfg
  - lshwres
- Password of said user.
- Location (FQDN/IP) of the HMC (e.g. `myhmc.mydomain`)
- HMC rest API base URL, for example:

```
https://myhmc.mydomain:12443
```

Modules generated by the plugin

The parameters monitored by the plugin are (grouped by element type):

- Current logical partitions
- Max logical partitions
- Max memory available
- Max memory installed
- Proc pool DefaultPool current proc units
- Proc pool DefaultPool max proc units
- Proc pool DevelopmentPool current proc units
- Proc pool DevelopmentPool max proc units
- Proc pool ProductionPool current proc units
- Proc pool ProductionPool max proc units
- Proc pool TestPool current proc units
- Proc pool TestPool max proc units
- Proc pool VIOPool current proc units

- Proc pool VIOPool max proc units
- Processor pools configured
- Processor units available
- Processor units installed
- State
- UUID
- Virtual proc units max

LPAR:

- Auto start: Configuration of autostart logical partitions.
- LPAR type: Type of logical partition.
- LPAR UUID : Used to query the HMC APIs.
- Max memory: Maximum memory.
- Max memory: Available memory.
- Processor units available: Processing units available.
- Processor units current : Installed processing units
- RMC IP address : RMC IP address.
- RMC state : RMC status in LPAR
- State : State of the logical partition.
- Virtual proc units : Virtual processing units assigned to this LPAR.

Virtual IO:

- Auto start: Configuration of autostart logical partitions.
- LPAR type: Type of logical partition.
- LPAR UUID : Used to query the HMC APIs.
- Max memory: Maximum memory.
- Max memory current: Available memory.
- Processor units available: Processing units available
- Processor units current : Installed processing units
- RMC IP address : RMC IP address.
- RMC state RMC : State of the RMC in LPAR.
- State : State of the logical partition.
- Virtual proc units : Virtual processing units assigned to this LPAR.

Configuring the IBM HMC plugin

Available configuration for the Pandora FMS plugin for IBM HMC:

Communication configuration towards the Pandora FMS server

- mode: Data transfer mode, “local” or “tentacle”.
- tentacle_ip: IP address of the Pandora FMS server, only applicable in tentacle mode.
- tentacle_port: Port on which the tentacle service is listening.
- tentacle_opts: Any extra options you have configured in the tentacle service.
- tentacle_client: Full path to your tentacle client.
- temp: Temporary working directory.
- local_folder: Delivery path for “local” data transfer mode.

HMC Access Settings

- hmc_host: IP or FQDN of the HMC.
- hmc_user: User with read permission.
- hmc_pass: Password.
- as_agent_plugin: The output of the plugin will be returned in XML format for executions scheduled with the Pandora FMS Agent (`as_agent_plugin = 1`). Or standard output (`as_agent_plugin = 0`) for executions scheduled with the system cron or performed as a server plugin.

Agent Settings

- agent_name: Optional, indicate a name for the parent Agent, by default `hostname`
- agent_interval: Agent interval, default 300.
- agent_group: Agent Group, by IBM default.

Customization of modules

- module_group: Group of Modules, by IBM default.
- module_interval: Module interval (multiplier), default 1.
- module_tags: Tags for the modules.

Entity renaming

To rename entities, block renaming is used:

```
 rename
 MyLPAR_NAME TO my new name
 MyLPAR_NAME2 TO my second new name
 rename_end
```

Running the IBM HMC plugin

The Pandora FMS plugin for monitoring IBM AIX systems through HMC is deployed as follows:

Setting the as_agent_plugin parameter to 1 (running as agent plugin):

```
module_plugin /usr/bin/perl pandora_hmc.pl pandora_hmc.conf
```

Setting the as_agent_plugin parameter to 0 (running as server plugin):

```
 # /etc/crontab
 */5 * * * * root /usr/bin/perl /root/hmc/pandora_hmc.pl
/root/vmware/pandora_hmc .conf
```

HPVM

How HPVM plugin works

This plugin allows you to monitor HPVM virtualization equipment. It is launched as an Agent plugin, generating in parallel one more agent for each hosted virtualized computerin the monitored system.

Local commands are used to collect information.

HPVM Plugin Requirements

Check each of the following steps:

- Deploy a Pandora FMS Agent on the computer you want to monitor.
- Have a user with permissions to run the plugin
- This user must have permissions to run the `hpvmstatus` command in order to interpret the output:

1. `hpvmstatus`.
2. `hpvmstatus -X`.
3. `hpvmstatus -r -X`.

HPVM plugin installation

Download a copy of the Pandora FMS plugin for HPVM HP Virtualization Manager monitoring from the module library. You can schedule the execution using collections and the deployed Pandora FMS Agent or extract the contents of the file to a non-volatile directory from where you can execute it through your system's cron.

```
unzip pandora_HPVM.zip
```

HPVM Plugin Configuration

Communication configuration towards the Pandora FMS server

- mode: Data transfer mode, “local” or “tentacle”.
- tentacle_ip: IP address of the Pandora FMS server, only applicable in tentacle mode.
- tentacle_port: Port on which the Tentacle service is listening.
- tentacle_opts: Any extra options you have configured in your Tentacle service.
- tentacle_client: Full path to your Tentacle client.
- temp: Temporary working directory.
- local_folder: Delivery path for “local” data transfer mode.

Agent Configuration

- agent_name: Optional, indicate a name for the parent Agent, by default `hostname`
- agent_interval: Agent interval, default 300.
- agent_group: Group to which the agents will belong, by default HPVM.

Customization of modules

- module_group: Module Group.
- module_interval: Module interval (multiplier), default 1.
- module_tags: Tags for the Modules.

Running the plugin

Running the plugin from the Pandora FMS Agent, it will appear in the Agent configuration file:

```
module_plugin /usr/bin/perl pandora_hpvm.pl pandora_hpvm.conf
```

For a manual test, configure the plugin by following the steps described, you can launch it as follows:

```
perl pandora_hpvm.pl pandora_hpvm.conf
```

Return to Pandora FMS documentation index