New Relic®

New Relic Unix Monitor

Install and Use Guide

System-Level Monitoring for AIX, Linux, OSX/MacOS & Solaris/SunOS

Harkamal Singh
New Relic Expert Services

Disclaimer

New Relic has this integration to enable monitoring of this technology. This integration is provided AS-IS WITHOUT WARRANTY OR SUPPORT, although you can report issues and contribute to this integration via GitHub. Support for this integration is available with an Expert Services subscription.

Document Change History

Version	Date	Change
0.2.5	2022-Mar-08	Added fedramp flag

Table of Contents

Requirements

- A New Relic account
- Supported Unix server to be monitor
- Java JRE/JDK v1.6 or later
- Network access to New Relic (proxies are supported, see details below)
- For Dashboard Installation: *curl* or *wget* installed

Supported Operating Systems

- AIX 7.x
- Linux - All sorts, including on ARM processors (such as Raspberry Pi) and z/Linux
- OSX / MacOS 10.9 ('Mavericks') and above
- Solaris/SunOS 10.x and 11.x

Installation & Usage Overview

1. Download the latest version of the agent.
2. Gunzip & untar on Unix server that you want to monitor
3. Configure and customize the following.
    a. Plugin configuration **required*

        plugin.json to set account ID, keys and static attributes

    b. Command configuration *(optional)*

        plugin-commands-.json files to configure additional Operating system commands*

    c. Startup script*(optional)*

        pluginctl.sh to set java and monitor path

4. Run "**./pluginctl.sh start"** from command-line to start the monitor
5. Check logs (in logs directory by default) for errors
6. Login to New Relic UI and find your data in Insights.
    a. In the data explorer, look for custom event types that start with "unixMonitor:"
    b. Possible event types (for out-of-the-box commands): unixMonitor:Disk, unixMonitor:DiskIO, unixMonitor:NetworkIO, unixMonitor:Process, unixMonitor:Stats, unixMonitor:Vmstat

Configuration

Plugin configuration

Plugin configuration is governed by "***plugin.json"*** file. This file is used to set global and agent settings for the monitor. A full example of the possible fields in *plugin.json* can also be found in *plugin-full-example.json*

Global settings

"global"

- OS (default: auto):
  Used to determine which commands to run and how to parse them. Leave set to auto to have the plugin figure that out (which normally works).
- account_id:
  New Relic account ID - the 6- or 7- digit number in the URL when you're logged into the account of your choosing.
- fedramp:
  A true or false string to indicate that the target is the New Relic Fedramp-authorized endpoint.

"infra_mode"

This object is to be defined when we use this monitor to push the data using new relic infrastructure agent

- "rpc_listener_port"
  RPC listener port on which is used to push the data infrastructure agent.

"insights_mode"

This object needs to be defined when we are directly posting to "Insights"

- insights_insert_key
  You must create an Insights Insert key, as described here.

"dashboards"[1]

This object needs to be defined if default dashboards need to be deployed using the pluginctl.sh script

- admin_api_key
  specify the Admin API key, as described here.
- integration_guid
  Default is UNIX.Infra.Monitor.
- dashboard_install
  Default is command_line.

Note: DO NOT DELETE OR CHANGE integration_guid AND dashboard_install UNLESS OTHERWISE INSTRUCTED. Both are required but must be left to their default values.

"proxy"

If using a proxy, the optional proxy object should be added to the global object in plugin.json.

- proxy_host
  Hostname for proxy
- proxy_port
  Proxy port
- proxy_username
  Proxy username
- proxy_password
  Proxy password

Agent settings

"agents"

- name
  If set to auto, the plugin will use that server's hostname. Otherwise, sets the hostname and agentName to whatever is set here.
- static (optional)
  An object containing static attributes (as name-value pairs) you want to appear in every event from this plugin

[1] ** *requires curl or wget installed*

Sample Plugin.json

```
{
          "global": {
              "OS": "auto",
              "account_id": "enter_NR_account_ID",
              "fedramp": "false",
              "insights_mode": {
                     "insights_insert_key": "enter_insights_insert_key"
              },
              "dashboards": {
                     "admin_api_key": "enter_admin_api_key",
                     "integration_guid": "UNIX.Infra.Monitor",
                     "dashboard_install": "command_line"
              },
              "proxy": {
                     "proxy_host": "enter_proxy_host",
                     "proxy_port": 5443,
                     "proxy_username": "enter_proxy_username",
                     "proxy_password": "enter_proxy_password"
              }
          },
          "agents": [
              {
                     "name": "auto",
                     "static": {
                            "attribute1": "attribute1_value",
                            "attribute2": 12345
                     }
              }
          ]
}
```

Command Configuration

Command configuration (plugin-commands-*. json) is used to configure the various commands that are used to collect the metrics from the target operating system. The kit contains default files for each supported Operating System (OS) configured to pull out the basic metrics. These files can be extended to pull additional metrics for the operating system by configuring additional commands in these files.

The files consist of the following parameters that can be configured in json format for a command in OS specific.

eventType

This attribute defines the type of event under which the metrics will be listed. The value listed here will be prepended with "unixMonitor:" and will listed in data explorer under custom event types. In the example configuration, the eventType of ***"File"*** is listed as ***unixMonitor:File*** under custom event types in Insights UI data explorer

command

This attribute defines the actual command that needs to be run to collect a metric. In example configuration the following command is run
"***ls -l /opt/New_Relic/newrelic-unix-monitor/config/plugin.json***"
The output from this command is parsed by the expression defined later in mapping section

```
$ ls -l /opt/New_Relic/newrelic-unix-monitor/config/plugin.json
-rw-r--r-- 1 user staff 355 Dec 19 07:22 /opt/New_Relic/newrelic-unix-
          monitor/config/plugin.json
```

checkAllRegex

This attribute can have two values (true/false). This governs if all the regular expressions listed under mappings will be matched. This is only for special cases where we need to iterate all the expressions. In example configuration this is set to false

lineLimit

This attribute defines how many lines need to be parsed. Default "0" will parse all the lines This is useful when we need to parse only a portion of command output for regular expression matching. In example configuration this is set to 0

interval

This attribute defines in minutes the interval between running the configured command. By default, interval 0, the commands would be run each time a harvest cycle is run. In example configuration this is set to 1440 i.e. 1day.

mappings

The mapping object defines how the output of the command needs to be read and what metrics they need to be matched to. Multiple mapping can be defined for a command. This object has sub attributes defined below

expression

This attribute defines the regular expression that is used to parse and match the line output. This expression needs to define the capturing group '()' to map them to metrics defined below. For more information refer java regular expression pattern matching [2]. In the example configuration below the following expression is used

```
(\\S+)\\s+(\\d+)\\s+(\\S+)\\s+(\\S+)\\s+(\\d+)\\s+(\\S+\\s+\\d+\\s+\\d+:\\d+)\\s+(\\S+)
```

This expression breaks down the output of the command into groups[3] as shown below in table

Match #	Group index	Start index	End index	Group content
1	0	0	112	-rw-r--r-- 1 user staff 355 Dec 19 07:22 /opt/New_Relic/newrelic-unix-monitor/config/plugin.json
1	1	0	10	-rw-r--r--
1	2	12	13	1
1	3	14	20	user
1	4	22	27	staff
1	5	29	32	355
1	6	33	45	Dec 19 07:22
1	7	46	112	/opt/New_Relic/newrelic-unix-monitor/config/plugin.json

[2] Java regex pattern matching @ https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html
[3] To parse the output for configuration use online java regex testers

metrics

This object defines the metrics that are parsed using the regular expression and are grouped. The metrics must be defined in sequential manner i.e. group 1 followed by group 2 and so on. The sub objects must define the two parameters listed below.

name

This attribute defines the name of the metric. In the example below for the first group name is listed as "***File.permissions***" the value that will map to this metric form the regular expression is ***"-rw-r--r—"***

type

This attribute defines the type of the metric. In most cases this will be listed as "**NORMAL**" although for complex cases "DELTA" is defined that reports the delta between the current and previous run of harvest cycle.

Example command configuration

```
{
          "eventType": "File",
          "command": "ls -l /opt/New_Relic/newrelic-unix-
          monitor/config/plugin.json",
          "checkAllRegex": false,
          "lineLimit": 0,
          "interval": 1440,
          "mappings":[
                    {
             "expression":
          "(\\S+)\\s+(\\d+)\\s+(\\S+)\\s+(\\S+)\\s+(\\d+)\\s+(\\S+\\s+\\d+\\s+\
          \d+:\\d+)\\s+(\\S+)",
                    "metrics": [
                              {
                                        "name": "File.permissions",
                                        "type": "NORMAL"
                              },
                              {
                                        "name": "File.links",
                                        "type": "NORMAL"
                              },
                              {
                                        "name": "File.owner",
                                        "type": "NORMAL"
                              },
                              {
                                        "name": "File.grouup",
                                        "type": "NORMAL"
                              },
                              {
                                        "name": "File.size",
                                        "type": "NORMAL"
                              },
                              {
                                        "name": "File.Date",
                                        "type": "NORMAL"
                              },
                              {
                                        "name": "File.Path",
                                        "type": "NORMAL"
                              }
                              ]
                    }
             ]
},
```

Startup script

The startup script "pluginctl.sh" is used to control the execution of the monitor, This script can also be modified to set various environmental parameters. The two most widely used configurations are

- PLUGIN_JAVA to set the location of Java on your server (including the "java" filename)
- PLUGIN_PATH to set the location of the Unix monitor

Usage

The startup script provides the following options.

```
Usage: ./pluginctl.sh [status|start|stop|stopremlogs|restart|dashboards]
```

status

Query the status of the monitor.

start

Start the monitor

stop

stop the monitor

stopremlogs

Stop the monitor and remove logs

restart

Restart the monitor

dashboards

Deploy dashboards. For more details check advanced configuration

Advanced configurations

Deploying Dashboards from separate server/desktop

You can initiate the dashboard install from a standalone machine (i.e. a tools server or your own mac, linux or cygwin laptop/desktop), you will need the following:

- pluginctl.sh
- config/plugin.json (including path) with the dashboard object filled

To install, run ./pluginctl.sh dashboards.

Fix for using the WebSphere JDK

If you are using the JDK that is packaged with WebSphere and see an exception in the logs like below, it is due to attempting to use the WebSphere SSL Factory instead of the IBM JSSE packages.

```
ERROR com.newrelic.metrics.publish.binding.Request - An error occurred
          communicating with the New Relic service
java.net.SocketException: java.lang.ClassNotFoundException: Cannot find
          the specified class
          com.ibm.websphere.ssl.protocol.SSLSocketFactory
```

If so, uncomment the following line in pluginctl.sh and restart the plugin.

```
# USE_IBM_JSSE=true
```

Fix for using Solaris 10

If you see the following error, it may be because the Bourne shell does not support certain syntax in the installer script.

```
pluginctl.sh: syntax error at line 240: `admin_api_key=$' unexpected
```

If so, use the Korn shell or bash (if available). Both were tested successfully in Solaris 10.