Plugin Monitoring for GLPI

Contents

Preface

Introduction

Introduction

Goals of this project

Description of the goals of this project

The goal of this plugin is to :

- Manage the configuration of Shinken monitoring via GLPI
- Get state of all services in real time in GLPI
- Get history of checks with RRDTOOL graphs
- Create tickets in GLPI helpdesk directly on event (planned)
- Use features of Shinken like Business Rules...

Why is it revolutionary?

A revolution?

We introduce new concepts into monitoring systems:

- Finish time you add each services on a device (computers, servers, switch...) and/or add into templates... **We now use INVENTORY of these devices to set automatically services** (so without human intervention, except for configuration on begin to install plugin Monitoring). This feature can be used with automatic inventory system like FusionInventory).

  

  Note: For example, all computers with name begin by "SRV" and have Apache installed, will be set check_http and check_apachestatus.

- We have defined templates for RRDTOOL in JSON format and can be used by other monitoring systems. We have created 2 kind of templates :
  - to convert perf_data of checks to RRDTOOL files
  - for graph generation based on these RRDTOOL files previously created and update with data

Links

How to reach the project

Internet site

Internet address of the project

You can find the projects' website on:

- *https://forge.indepnet.net/projects/monitoring/*

RRDTOOL templates can be found at this address:

- *https://github.com/rrdtooltemplates/rrdtool_templates*

IRC-Channel

Address of the irc-Channel

Here is the information to find the projects' IRC channel:

- Server: *freenode.net*
- Channel: *#glpimonitoring*
- Ask for user *ddurieux*

Terms used in this solution

Some terms has been better defined

We have redefined term to be more easier to use.

Monitoring terms are in italic and plugin monitoring terms are in bold:

- *Services templates* => **Components**
- *Services* => **Resources**
- *Timeperiods* => **Agendas/Calendar**
- *Business Rules* => **Services catalog**

Part

I

Preparation

Topics:

Elements needed to prepare the solution

This chapter details the preparation of installation and usage of this solution.

Required tools

List and description of tools required

This is the list of tools required for the solution :

GLPI (Gestion Libre de Parc Informatique)

GLPI is an open source tool for asset management and helpdesk.

License: GPLv2

Web site: *http://www.glpi-project.org/*

Version required: 0.80.x (if possible the latest version available: 0.80.7)

There are many plugins available to extend the available features.

FusionInventory

FusionInventory is a tool to do inventory of computers, servers, switches, network printers...

Web site: *http://www.fusioninventory.org/*

FusionInventory is consisting of 2 parts, an agent:

License: GPLv2

Version required: 2.1.x (if possible the latest version available: 2.1.14)

and a plugin for GLPI (server side):

License: AGPLv3

Version required: 0.80+1.1

Webservices plugin for GLPI

It's a GLPI plugin add possibilities to access to GLPI by webservice (XML-RPC and SOAP)

License: GPLv2

Web site: *https://forge.indepnet.net/projects/webservices*

Version required: 1.2.0

Shinken

Shinken is a monitoring tool (fork of Nagios) written in Python. Shinken will be doing the checks on the resources.

License: AGPLv3

Web site: *http://www.shinken-monitoring.org/*

Version required: 1.0.1

Monitoring plugin for GLPI

It's this plugin that will manage Shinken monitoring from inside GLPI (and possibly other monitoring systems in future).

License: AGPLv3

Web site: *https://forge.indepnet.net/projects/monitoring/*

Version required: 0.80+1.2

Android Application Monitoring for GLPI

How this solution works

Not to be confused, this is a description of how the solution works

Legend:

- *[D]* = automatic operation
- *[M]* = manual operation
- *[DM]* = can be do dynamically or manually

This is the description of the processes:

- *[M]* Technician installs a new server
- *[DM]* Installation of FusionInventory agent to have a complete inventory of this server
- *[D]* FusionInventory sends an inventory to GLPI
- *[D]* Plugin Monitoring set resources to this new server defined by rules (in components catalogs)
- *[DM]* Restart Shinken' Arbiter module to load new configuration
- *[D]* Shinken runs with new checks
- *[D]* Shinken send events to GLPI MySQL database
- *[D]* A system cron script updates RRDTOOL files with these events and generates/updates the graphs
- User can see events in GLPI or via an application on his Android smartphone

Conclusion: Adding a new device to be monitored is simple and doesn't require any human intervention.

Part
II

Installation

Topics:

Installation of all elements/tools

Follow this order to get the solution up and running:

Shinken

Installation of Shinken

For a basic installation 10 minutes installation of Shinken, see the official documentation on:*http://www.shinken-monitoring.org/wiki/shinken_10min_start*

The version 2.6 or 2.7 of Python is recommanded for Shinken.

GLPI

Installation of GLPI

For the installation of GLPI in your environment, have a look at the official GLPI documentation at: h*ttps://forge.indepnet.net/projects/glpidoc/files*

FusionInventory

Installation of FusionInventory

You'll then need to install and configure a FusionInventory environment (server and agent), for this, have a look at the FusionInventory documentation located at: *http://fusioninventory.org/wordpress/documentation/*

WebServices plugin for GLPI

Installation of Webservices plugin

- Get the latest version for your version of GLPI: *https://forge.indepnet.net/projects/webservices/files*
- Decompress the downloaded archive into the <glpiFolder>/plugins/
- Login to your GLPI server (web)
- Go to *Setup* > *Plugins*
- Click on *install* to integrate the WebServices plugin into GLPI
- Click on *activate* to enable the WebServices plugin
- The plugin is installed

Monitoring plugin for GLPI

Installation of Monitoring plugin

- Get the latest version for your version of GLPI: *https://forge.indepnet.net/projects/monitoring/files*
- Decompress the downloaded archive into the <glpiFolder>/plugins/
- Login to your GLPI server (web)
- Go to *Setup* > *Plugins*
- Click on *install* to integrate the monitoring plugin into GLPI
- Click on *activate* to enable the monitoring plugin
- The plugin is installed

Android: glpi_monitoring app (optional)

Installation of application glpi_monitoring for android

- Download the latest APK file with the Android browser: *https://forge.indepnet.net/projects/monitoring/files*
- Open the apk package and install the app.

- The application is installed

Part III

Configuration

Topics:

Configuration of all elements/tools

This chapter explains how to configure the components.

Manage TAGS

Manage TAGS

Important: In most cases, you don't need use TAGS, so leave empty in configuration.

In some cases, you want to have a Shinken server related to a GLPI entity (so have 2 Shinken servers related to 2 different entities). This configuration is used when each entity is a different and distant site and not have link between these sites, but they have link to the GLPI (hosted on a web server for example). So in this case, we must tag the GLPI entities and each Shinken server.

- In Shinken, it's in configuration of Arbiter module named *GLPI*.
- In GLPI, it's on tab *Monitoring* in entities forms.

Create GLPI accounts

Create a GLPI account

You need to create GLPI accounts:

- Shinken needs a GLPI account to connect via the WebServices plugin to get its configuration. Create a local GLPI account via *Administration* > *User*.

  For example:

  - Login: *shinken*
  - Password: *passshinken*
- Normal user account(s): Don't use GLPI's default super-admin, create your own account (or import from LDAP, see the GLPI documentation for this).

GLPI webservice

Configure GLPI webservice

For Shinken

Configure GLPI Web Services for Shinken

In GLPI, go to the menu *Plugins* > *Web Services*, and add a new web service. Set these values:

- Name: *Shinken*
- Enabled Services: *Yes*
- Compression enabled: *No* (not tested with compression enabled)
- Log connections: *No* (activate it if you want to keep track of connections)
- Debug: *No* (enable for debugging)
- SQL pattern for services: .*
- IP address range: set IP range, in this case set twice the IP of your server where Shinken is installed
- User name: (keep empty in this case)
- Password: (keep empty in this case)

Click on the add button. The web service for Shinken is then created and access is granted to the Shinken server.

For the Android application

Configure GLPI webservice for the Android application

Configure this only if you want to use Android application.

In GLPI, go to menu *Plugins* > *Web Services*, and add a new web service. Set these values:

- Name: *Android glpi_monitoring*
- Enabled services: *Yes*
- Compression enabled: *No* (not tested with compression enabled)
- Log connections: *No* (activate it if you want to keep track of connections)
- Debug: *No* (enable for debugging)
- SQL pattern for services: *monitoring.**
- IP address range: if you have a fixed IP on your Android phone (not likely), set this IP in the 2 inputs, else you can set a big range like *1.0.0.1* to *254.254.254.254*
- User name: (keep empty in this case)
- Password: (keep empty in this case)

Click on the add button: The webservice for the Android application is created and granted to access GLPI.

Create MySQL accounts

Create a MySQL account

You need to create a MySQL account because the *Broker* module of Shinken will add and update events in the database.

For example create :

- Login: *shinkenbroker*
- Password: *passshinken*
- Host: IP of your server where Shinken is installed

You can set rights to all GLPI database for this account but, for more security, give right only to tables

- glpi database\glpi_plugin_monitoring_services (SELECT and UPDATE only)
- glpi database\glpi_plugin_monitoring_serviceevents (INSERT only)
- glpi database\glpi_plugin_monitoring_servicescatalogs (SELECT and UPDATE only)

Shinken

Configure Shinken

Arbiter module

Configure Shinken' Arbiter module

The *Arbiter* module is used to load Shinken' configuration. Edit the configuration file *shinken-specific.cfg* and modify this module:

```
define module{
       module_name GLPI
       module_type glpi
       uri http://localhost/glpi/plugins/webservices/xmlrpc.php
       login_name glpi
       login_password glpi
       tag
}
```

The default values to change for your environment are:

- uri: url of GLPI, finish always by */plugins/webservices/xmlrpc.php*
- login_name: GLPI account created in "Create GLPI accounts" (in our example, it's *shinken*)
- login_password: password of the GLPI account (in our example, it's *passshinken*)

- tag: define tag if you use it, else leave it empty

The module is now defined, then add it in arbiter configuration:

```
define arbiter{
       arbiter_name Arbiter-Master
# host_name node1 ;result of the hostname command under Unix
       address localhost ;IP or DNS address
       port 7770
       spare 0
# uncomment the line below if you want to use some modules for your arbiter
# modules CommandFile, Mongodb, NSCA, VMWare_auto_linking

# List of interesting modules :
# CommandFile : open the named pipe nagios.cmd
# Mongodb : load hosts from a mongodb database
# PickleRetentionArbiter : save data before exiting
# NSCA : NSCA server
# VMWare_auto_linking : lookup at Vphere server for dependencies
# GLPI : import hosts from GLPI
# TSCA : TSCA server
[...]
```

So add module and we will have that :

```
define arbiter{
       arbiter_name Arbiter-Master
# host_name node1 ;result of the hostname command under Unix
       address localhost ;IP or DNS address
       port 7770
       spare 0
# uncomment the line below if you want to use some modules for your arbiter
# modules CommandFile, Mongodb, NSCA, VMWare_auto_linking

modules GLPI

# List of interesting modules :
# CommandFile : open the named pipe nagios.cmd
# Mongodb : load hosts from a mongodb database
# PickleRetentionArbiter : save data before exiting
# NSCA : NSCA server
# VMWare_auto_linking : lookup at Vphere server for dependencies
# GLPI : import hosts from GLPI
# TSCA : TSCA server
[...]
```

Broker module

Configure Shinken' broker module

The Broker module is used to send events. Edit the configuration file *shinken-specific.cfg* and modify this module:

```
# send into GLPI DB
# =============== Work with Plugin Monitoring of GLPI ===============
define module{
  module_name glpidb
  module_type glpidb
  database glpi ; database name
  user root ; database user
  password root ; must be changed
  host localhost ; host to connect to
```

```
}
```

The values to change here are:

- database: name of GLPI MySQL database
- user: MySQL account created on chapter *Create MySQL accounts* on page 17 (in our example, it's *shinkenbroker*)
- password: password of GLPI account (in our example, it's *passshinken*)
- host: IP or name of server where MySQL server is installed.

The module is defined, now add it in Arbiter configuration:

```
#The broker manages data export (in flat file or in database)
#with its modules
#Here just log files and status.dat file modules
define broker{
       broker_name broker-1
       address localhost
       port 7772
       spare 0

       # Which modules to load? LiveSatus and logs by default.
       modules Livestatus, Simple-log, WebUI



# Other interesting modules to add :
# PickleRetentionBroker : save data when quitting
# ToNdodb_Mysql : NDO database support
# NPCDMOD : Use the PNP addon
# Graphite-Perfdata : Use teh Graphite backend for perfdata
# WebUI : Shinken Web interface
[...]
```

So add module and we will have that :

```
#The broker manages data export (in flat file or in database)
#with its modules
#Here just log files and status.dat file modules
define broker{
       broker_name broker-1
       address localhost
       port 7772
       spare 0

       # Which modules to load? LiveSatus and logs by default.
       modules Livestatus, Simple-log, WebUI, glpidb



# Other interesting modules to add :
# PickleRetentionBroker : save data when quitting
# ToNdodb_Mysql : NDO database support
# NPCDMOD : Use the PNP addon
# Graphite-Perfdata : Use teh Graphite backend for perfdata
# WebUI : Shinken Web interface
[...]
```

In a simple implementation, Livestatus, Simple-log and WebUI can be remove on load on line begin by *modules*.

Common objects

Configure common objects of Shinken

By default, Shinken load common objects like commands, timeperiod, hosts, services. If you want to only load configuration with GLPI, you may modify file *nagios.cfg*:

```
#Configuration files with common objects like commands, timeperiods,
#or templates that are used by the host/service/contacts
#cfg_file=commons.cfg
cfg_file=commands.cfg
cfg_file=timeperiods.cfg
cfg_file=escalations.cfg
cfg_file=dependencies.cfg
cfg_file=contacts.cfg
#cfg_dir=/opt/omd/sites/pilot/etc/nagios/conf.d/pilot/dynamic

#Now templates of hosts, services and contacts
cfg_file=templates.cfg

#Now groups
cfg_file=servicegroups.cfg
cfg_file=hostgroups.cfg
cfg_file=contactgroups.cfg

#and now real hosts, services, packs and
# discovered hosts
cfg_dir=hosts
cfg_dir=services
cfg_dir=packs
cfg_dir=objects/discovery

# Un comment this for a sample configuration
#cfg_dir=sample
[...]
```

By

```
#Configuration files with common objects like commands, timeperiods,
#or templates that are used by the host/service/contacts
#cfg_file=commons.cfg
#cfg_file=commands.cfg
#cfg_file=timeperiods.cfg
#cfg_file=escalations.cfg
#cfg_file=dependencies.cfg
#cfg_file=contacts.cfg
#cfg_dir=/opt/omd/sites/pilot/etc/nagios/conf.d/pilot/dynamic

#Now templates of hosts, services and contacts
#cfg_file=templates.cfg

#Now groups
#cfg_file=servicegroups.cfg
#cfg_file=hostgroups.cfg
#cfg_file=contactgroups.cfg

#and now real hosts, services, packs and
# discovered hosts
#cfg_dir=hosts
#cfg_dir=services
#cfg_dir=packs
#cfg_dir=objects/discovery
```

```
# Un comment this for a sample configuration
#cfg_dir=sample
[...]
```

Plugin monitoring

Configure the plugin monitoring

Configure checks definitions

Configuration of checks

This definition of checks is used to define to a resource, what kind of check (all 5 minutes with 3 retries, all 15 minutes with 2 retries...).

To access to this menu, go on *Plugins* > *Monitoring* > *Check definition*.

Elements to configure for each definition check are:

- *Name*: give a name.
- *Time in minutes between 2 checks*: define the time in minutes between 2 checks.
- *Max check attempts (number of retries)*: define number of retries when a state (OK, warning, critical...) change before get really this new state.
- *Time in minutes between 2 retry*: time between 2 retries.

Define commands

Configuration of commands

These commands are used for checks and detect state of service to monitor.

Some commands are added when install the plugin.

To access to this menu, go on *Plugins* > *Monitoring* > *Commands*

Elements to configure for each command check are:

- *Name*: give a name
- *Command name*: define command name, must be unique and without special characters. It's name of script Shinken will execute to verify you want (disk, cpu, memory...).
- *Command line*: path and file name to run (on server where Shinken is installed because it's Shinken will run this command). You can use Shinken Macro like *$PLUGINSDIR$*, *$MYSQLUSER$*... and you can use macro for dynamic arguments used into plugin_monitoring like *$ARG1$*, *$ARG2$*, *$ARG3$*...
- Arguments description: this is visible only when have *$ARG1$*, *$ARG2$*, *$ARG3$*... in *Command line* field. For each arguments, have an input with description of this argument. For example, if have in *Command line* the command *$PLUGINSDIR$/check_load -r -w $ARG1$ -c $ARG2$*, will have 2 input for each of the 2 arguments and in *ARG1* set what is this argument is for: *WARNING status if load average exceeds WLOADn (WLOAD1,WLOAD5,WLOAD15)*. Define these arguments is important for define components, because say what for this argument will be used.

Define agenda

Configuration of agenda

In this plugin, some configurations will use agenda to define:

- Check period (when checks will be executed: all days 24h/24, only works hours from Monday to Friday....)
- Notification period (when notifications will be sent: all days 24h/24, only works hours from Monday to Friday....)

To access to this menu, go on *Plugins* > *Monitoring* > *Calendar*

You create agendas and set days and hours when action will be executed.

Host configuration

Configuration of hosts

For each host you will monitor, you must define:

- Command: check to verify host is alive
- Check definition
- Check period (when checks will be executed: all days 24h/24, only works hours from Monday to Friday....)

There are 2 possibilities to configure:

- Define on entity for all hosts: see that on tab *Monitoring* on entity form. All hosts on this entity will get these values.
- Define on each host: see that on tab *Monitoring - Resources* on each host form (computer, network equipment, printer... form).

Note: By default, configuration is set (on install or update plugin) on root entity.

RRDTOOL templates

Manage RRDTOOL templates

If you want to have/generate RRDTOOL graphs of resources, templates will be uploaded in plugin. You can access to this menu by *Plugins* > *Monitoring* > *RRDTOOL templates*.

The templates can be downloaded in GIT repository *https://github.com/rrdtooltemplates/rrdtool_templates*.

For the generation of graphs, 2 types of templates will be present:

- Perfdata: used to parse event and put in RRDTOOL file.
- Graph: used to generate graph with data in the RRDTOOL file.

Contact templates

Manage contact templates

You can manage notifications templates (contact templates) to configure how and when send mails on critical check.

You can access to this menu by *Plugins* > *Monitoring* > *Contact templates*.

Note: You must create at least one *default* template.

After have created templates, you can assign a template to an user account (if you don't want using the default template for this user). For this, see tab *Monitoring-Contact* in a user form (menu *Administration* > *Users*).

Define components

Configuration of components

These components are used to define what monitor for each hosts.

To access to this menu, go on *Plugins* > *Monitoring* > *Components*

Elements to configure for each component check are:

- *Name*: give a name
- *Command*: define command (see chapter *configuration_pluginmonitoring_command.dita*). This field is required except if you define this component as a remote check.
- *Template (for graphs generation)*: select a template for graph generation (RRDTOOL).
- *Check definition*: define the definition check for this component (see chapter *Configure checks definitions* on page 21).
- *Active checks*: is this component is an active check (Shinken run the command)

- *Passive check*: is this component is a passive check (remote host run the command and send event to Shinken only when change)
- *Check period*: set period (it's GLPI calendar) to define when check is running (24x7, not week end....)

It's possible to define a remote check, remote server run itself command and return to Shinken the result). This is the fields to set for this:

- *Utility used for remote check*: choose system to use: *byssh*, *nrpe* or *nsca*.
- *Use arguments (Only for NRPE)*: if *nrpe* is selected, it's possible to define the command to run, so defined here and not in nrpe.cfg on remote system. **Active this is dangerous for security!**
- *Alias command if required (Only for NRPE)*: set the command defined in nrpe.cfg on remote servers.

Define components catalogs

Configuration of components catalogs

These components catalogs are used to define for what hosts monitoring system will check resources.

To access to this menu, go on *Plugins* > *Monitoring* > *Components catalog*

Components

Define components of *components catalogs*

In this tab, there is the list of components for hosts of this components catalog. Simply add components.

For example: For a *components catalog* called 'http servers', it can have components *check_http* and *check_apachestatus*

Hosts

Add hosts to *components catalogs*

There are two methods to add hosts in a components catalog.

Static hosts

Add static hosts to *components catalogs*

The first method to add hosts is to add them manually by a user. This method is called *static hosts* and available in the tab have same name.

Dynamic hosts

Configure dynamic hosts to *components catalogs*

The second method to add hosts is to define rules (search). This method is called *dynamic hosts* and available in the tabs *rules* and *dynamic hosts*.

In tab *rules*, it's possible to add **only one rule** by item type. These rules use GLPI search engine. If a computer rule is defined by *Operating system* contains *Linux*, all computers match this search query will be added automatically in tab *dynamic hosts*.

Note: When add or modify an item type (like a Computer in our example), the rule will replay and update dynamic hosts list without human intervention. So when add or update an inventory with automatic tool FusionInventory, it will update hosts and components affected.

Contacts

Define contacts of *components catalogs*

In this tab, there is the list of contact or contacts groups. It's used by Shinken to send notification in case a resource become critical. So these contacts will be associated to components to hosts configured in this catalog.

Define services catalogs

Configuration of services catalogs

The services catalog is used to check services (our web application is OK? Is one of the 3 Windows domain controller is OK?). It is based on resources yet defined by components catalogs. Better is to see with an example.

Our GLPI application is composed by:

- 3 Apache servers in load balancing
- 1 MySQL server

We create a service catalog with 3 groups (operator between the groups is: *and*):

- Group Apache

  We define logical operator *or*, this say one of the 3 Apache servers must be OK. We add the 3 resources:

  - *Check_http* on *serverA1*
  - *Check_http* on *serverA2*
  - *Check_http* on *serverA3*
- Group MySQL

  We define logical operator *or*, but we have only one MySQL server, so it may be always ok. We add the resource:

  - *Check_mysql* on *serverM1*
- Group Responses times/scenario

  We define logical operator *or*, but we have only one check on test response time of our GLPI application and can test in same time if we can log in, go on computer list page for example with JMETER.

  - Check_jmeter (attached on server you want, so it's defined on a components catalog). This check define a critical when responsive time if more than 50ms and if it can't log in GLPI application (scenario).

Now it's configured, see the cases Shinken return us a critical state:

- The 3 Apache servers are stopped (if 2 are stopped, it's always ok)
- MySQL server is down
- Response time of GLPI is more than 50ms
- Scenario to log in GLPI isn't ok

Part
IV

Daily use

Topics:

Daily use of Monitoring system

This chapter explains how to configure the components.

Reload Shinken

Reload Shinken

When add or modify device (device configuration), configuration is modified in GLPI. Restart Shinken's Arbiter is required to reload this new configuration. It can be set by user with restart command or perhaps by a script. We working with Shinken Team to be possible to reload it from GLPI manually or dynamically when configuration change.

Display events

Display events

Where see the events?

In GLPI

In GLPI

Events can be seen in 2 places in GLPI:

- General dashboard in menu *Plugins* > *Monitoring* > *Dashboard*. There are different views:
    - *Services catalog*: display state of services catalogs
    - *Components catalog*: display state of components catalogs
    - *All resources*: display all resources and you can use search engine to find a resource or a type of resource.
- In each device form, a tab *Monitoring-resources* display resources and state of this device.

If a template has been defined on components, graphs will be available on display.

In the Android application

In the Android application

Display number of devices in state Critical, warning and OK (type of view *Components catalog* or *All resources* can be set in *options* menu.

When release the button with color (green, orange or red) a new window appear with list of resources/host with this state.