


 BaldMansMojo / [check_vmware_esx](#)


check_vmware_esx Fork of check_vmware_api.pl

 GPL-2.0 License 105 stars  55 forks Star Watch Code Issues 6 Pull requests 7 Actions Projects Security In

Join GitHub today

Dismiss

GitHub is home to over 50 million developers working together to host and review code, manage projects, and build software together.

[Sign up](#) master ▾

BaldMansMojo committed on Nov 26, 2019 ...

 160 commits[View code](#)

README.md

check_vmware_esx for VMware Monitoring

Table of Contents

1. [About](#)
2. [License](#)
3. [Support](#)
4. [Requirements](#)
5. [Installation](#)

- 6. [Configuration](#)
- 7. [FAQ](#)
- 8. [Thanks](#)
- 9. [Contributing](#)

About

Check VMware ESX is a plugin to monitor VMware ESX and vSphere servers. You can monitor either a single ESX(i)/vSphere server or a VMware VirtualCenter/vCenter Server and individual virtual machines. If you have a VMware cluster you should monitor the data center (VMware VirtualCenter/vCenter Server) and not the individual ESX/vSphere servers.

Supports check status of ESX(i) server, vSphere 5.1 up to vSphere 6.5.

chech_vmware_esx is a fork of [check_vmware_api](#).

License

This plugin is licensed under the terms of the GNU General Public License Version 2, you will find a copy of this license in the LICENSE file included in the source package.

Support

Join the community channels from Icinga, Nagios, Naemon for questions. A common place to kindly ask and discuss can be joined at [monitoring-portal.org](#).

Requirements

VMware Perl SDK

Download the [VMware Perl SDK](#) (requires a free account that you sign up for at the same page).

Do not use SDK 6.0 because it contains some bugs and incompatibilities. Use SDK 5.5 or 6.5. Both work well.

Install the SDK. An installation guide can be found [here](#).

Perl Modules

Please ensure that the following Perl modules are installed on your system:

- `File::Basename`
- `HTTP::Date`
- `Getopt::Long`
- `Time::Duration`
- `Time::HiRes`
- `VMware::VIRuntime` (see above)

Installation

Best practice is to keep your own plugins separated from the official Monitoring Plugins. You can adopt the paths in the following sections to your own needs. For simplicity, the documentation refers to the existing Monitoring Plugins directory where Monitoring Cores would expect them.

Single File Installation

Similar to how `check_mysql_health` and variants are compiled into a single Perl plugin script, Sven Nierlein contributed a Makefile which generates a single file to install.

Example on Debian/Ubuntu:

```
make all
cp check_vmware_esx /usr/lib/nagios/plugins/
chmod 755 /usr/lib/nagios/plugins/check_vmware_esx
```

Note

This is recommended if you are having trouble with missing Perl modules, e.g. `help.pm` is not found.

Now proceed with the [configuration](#) chapter.

Modular Installation

The plugin script requires the `modules` directory being copied into an upgrade safe location. Best is to put them outside of the system's Perl path. The default settings expects them in the same directory where the plugin is copied to, e.g. `/usr/lib/nagios/plugins` on Debian/Ubuntu/SUSE or `/usr/lib64/nagios/plugins` on RHEL/CentOS.

Example on Debian/Ubuntu:

```
cp -r modules /usr/lib/nagios/plugins/  
cp check_vmware_esx.pl /usr/lib/nagios/plugins/check_vmware_esx  
chmod 755 /usr/lib/nagios/plugins/check_vmware_esx
```

In case you want to put `modules` into a different location, you need to edit the plugin script and add your own modules path, e.g. `/usr/lib/nagios/vmware/modules`.

```
vim /usr/lib/nagios/plugins/check_vmware_esx  
  
#use lib "modules";  
use lib "/usr/lib/nagios/vmware/modules";
```

Now proceed with the [configuration](#) chapter.

Configuration

Evaluate the required parameters and modules by testing the plugin as nagios/icinga user on the shell. Invoke `--help` to find out more about possible modes and parameters.

Monitoring Configuration

Icinga 2 provides [VMware CheckCommand definitions](#) in its template library already. Icinga 1.x/Nagios require you to create command definitions beforehand.

Example for Icinga 2 and DC volumes:

```
apply Service "vcenter-volumes" {  
    check_command = "vmware-esx-dc-volumes"  
    vars.vmware_datacenter = "vcenter.yourdomain.local"  
    vars.vmware_username = "icinga@yourdomain.local"  
    vars.vmware_password = "password"  
  
    assign where "VMware" in host.groups  
}
```

Authentication

In order to avoid credentials in your service definition, you can also use an authorization file and pass it as argument to the plugin.

```
mkdir -p /etc/icinga2/authfiles/vcenter-auth
cat >/etc/icinga2/authfiles/vcenter-auth <<EOF
username=username@yourdomain.com
password=mypassword
EOF
```

Example for Icinga 2 and DC volumes:

```
apply Service "vcenter-volumes" {
    check_command = "vmware-esx-dc-volumes"
    vars.vmware_datacenter = "vcenter.yourdomain.local"
    vars.vmware_authfile = "/etc/icinga2/authfiles/vcenter-auth"

    assign where "VMware" in host.groups
}
```

Session File

Adjust the path for `$sessionfile_dir_def` inside the script if required. This defaults to `/tmp`.

FAQ

Timeouts

If you are encountering timeouts using the VMware Perl SDK, check [this blog entry](#).

You then need to downgrade the `libwww-perl` package.

Compatibility with Plugin Developer Guidelines

According to the guidelines every plugin should start its output with

SERVICE STATUS: First line of output | First part of performance data

The service status part wasn't implemented in output in the past because from a technically view it is not necessary. For notifications this string is available as a macro (\$SERVICESTATE\$) together with the state ID (\$SERVICESTATEID\$). If the macro is used for notifications you will have a doubled string (for example CRITICAL CRITICAL). While this is only cosmetics for email and in the webgui (if a check is red you need the extra string CRITICAL only when you are clour-blind) it can have side effects using SMS because the string length in SMS is limited. So you have to filter the message (sed) before you send it.

To fulfill the rules of the Plugin Developer Guidelines an extra option --statelabels=<y/n> was added. The default is set in the plugin in the variable \$statelabels_def which is set to "y" by default.

Simon Meggle proposed --nostatelabels (thanks Simon - it was a good idea to add it) but I preferred it the other way round first. Caused by some feedback I decided to offer all options. If you don't like statelabels (like me) just set the variable to "n".

HTML in Output

HTML in output is caused by the option --multiline or in some situations where you need a multiline output in the service overview window. So here a tag is used to have a line break. Using --multiline will always cause a instead of \n. (Remember - having a standard multiline output from a plugin only the first line will be displayed in the overview window. The rest is with the details view).

The HTML tags must be filtered out before using the output in notifications like an email. sed will do the job:

```
sed 's/<[Bb][Rr]>/&\n/g' | sed 's/<[^<>]*>//g'
```

Example for Nagios/Icinga 1.x:

```
# 'notify-by-email' command definition
define command{
    command_name    notify-by-email
    command_line    /usr/bin/printf "%b" "Message from
Nagios:\n\nNotification Type: $NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nHost:
$HOSTNAME$\nHostalias: $HOSTALIAS$\nAddress: $HOSTADDRESS$\nState:
$SERVICESTATE$\n\nDate/Time: $SHORTDATETIME$\n\nAdditional
Info:\n\n$SERVICEOUTPUT$\n$LONGSERVICEOUTPUT$" | sed 's/<[Bb][Rr]>/&\n/g' | sed
's/<[^<>]*>//g' | /bin/mail -s "*** $NOTIFICATIONTYPE$ alert -
$HOSTNAME$/$SERVICEDESC$ is $SERVICESTATE$ ***" $CONTACTEMAIL$
}
```

If you are using Icinga 2, you need to adjust the notification scripts.

Using Open VM Tools

VMware recommends using the Open VM Tools (<https://github.com/vmware/open-vm-tools>) instead of host deployed tools. According to pkgs.org (<http://pkgs.org/download/open-vm-tools>) these tools are available for most Linux systems. For SLES you find them via https://www.suse.com/communities/conversations/free_tools/installing-vmware-tools-easy-way-osp/.

"VMware Tools is installed, but it is not managed by VMware" is now quite ok for package based Linux installation, but for all other systems (like MS Windows) this will cause a warning because it's not ok.

Therefore I implemented a new option --open-vm-tools.

But how to handle it in a command definition? Two definitions (one for Linux, one for the rest?)

No - we handle this option over to the command as an argument:

```
define command{
    command_name    check_vshpere_tools_state
    command_line    /MyPluginPath/check_vmware_esx.pl -D
$DATACENTER_HOSTADDRESS$ -u $ARG1$ -p $ARG2$ -S runtime -s tools $ARG3$
}
```

Service check for Linux:

```
define service{
    ....
    .
    .
    .
    .....
    check_command
check_vsphere_tools_state!Nagios!MyMonitorPassword!"--open-vm-tools"
}
```

Service check for the rest:

```
define service{
    ....
```

```

.
.
.
.....
check_command
check_vsphere_tools_state!Nagios!MyMonitorPassword
}

```

So you can see in the second definition `$ARG3$` is empty and therefore the option is not set.

Using a session file

To reduce amounts of login/logout events in the vSphere logfiles or a lot of open sessions using sessionfiles the login part has been totally rewritten with version 0.9.8.

Using session files is now the default. Only one session file per host or vCenter is used as default. The sessionfile name is automatically set to the vSphere host or the vCenter (IP or name - whatever is used in the check). The file will only update if the session gets invalidated at some point.

Multiple sessions are possible using different session file names. To form different session file names the default name is enhanced by the value you set with `--sessionfile`.

`--sessionfile` is now optional and only used to enhance the sessionfile name to have multiple sessions.

Example command and service check definition:

```

define command{
    command_name      check_vsphere_health
    command_line      /MyPluginPath/check_vmware_esx.pl -H $HOSTADDRESS$ -u
$ARG1$ -p $ARG2$ -S runtime -s health
}

define service{
    active_checks_enabled      1
    passive_checks_enabled     1
    parallelize_check          1
    obsess_over_service        1
    check_freshness            0
    notifications_enabled      1
    event_handler_enabled      1
    flap_detection_enabled     1
    process_perf_data          0
    retain_status_information   1
}

```



```

    retain_nonstatus_information    1

    host_name                      vmsrv1
    service_description            Health
    is_volatile                    0
    check_period                   24x7
    max_check_attempts             5
    normal_check_interval          5
    retry_check_interval           1
    contact_groups                 sysadmins
    notification_interval          1440
    notification_period            24x7
    notification_options           c,w,r
    check_command
check_vsphere_health!Nagios!MyMonitorPassword
}

```

Example for session file name and lock file name:

```

192.168.10.12_session (default)
192.168.10.12_session_locked (default)

```

Example caommand and service check definition (enhenced):

```

define command{
    command_name    check_vsphere_health
    command_line    /MyPluginPath/check_vmware_esx.pl -H $HOSTADDRESS$ -u
$ARG1$ -p $ARG2$ -S runtime -s health --sessionfile=$ARG3$
}

define service{
    active_checks_enabled      1
    passive_checks_enabled     1
    parallelize_check          1
    obsess_over_service        1
    check_freshness            0
    notifications_enabled      1
    event_handler_enabled      1
    flap_detection_enabled     1
    process_perf_data          0
    retain_status_information   1
    retain_nonstatus_information 1

    host_name                      vmsrv1
    service_description            Health
    is_volatile                    0
    check_period                   24x7

```

```

max_check_attempts      5
normal_check_interval   5
retry_check_interval    1
contact_groups          sysadmins
notification_interval   1440
notification_period     24x7
notification_options    c,w,r
check_command
check_vsphere_health!Nagios!MyMonitorPassword!healthchecks
}

```

Example for session file name and lock file name:

```

192.168.10.12_healthchecks_session (enhenced)
192.168.10.12_healthchecks_session_locked (enhenced)

```

Optional a path different from the default one can be set with `--sessionfilename=<directory> .`

Storage

The module `host_storage_info.pm` (contains `host_storage_info()`) is a extensive rewrite. Most of the code was rewritten. It now tested with iSCSI by customers and it is reported to run without problems.

Multipath:

The original version was as buggy and misleading as a piece of code can be. A threshold here was absolute nonsense. Why? At top level is a LUN which has a SCSI-ID. This LUN is connected to the storage in a SAN with a virtual path called a multipath because it consists of several physical connections(physical paths). If you break down the data delivered by VMware you will see that in the section for each physical path the associated multipath with it's state was also listed. `check_vmware_api.pl` (and `check_esx3.pl`) counted all those multipath entry. For example if you had 4 physical paths for one multipath the counted 4 multipaths and checked the multipath state. This was absolute nonsense because (under normal conditions) a multipath state is only dead if all physical paths (4 in the example) are dead.

The new check checks the state of the multipath AND the state of the physical paths. So for example if you have two parallel FC switches in a multipath environment and one is dead the multipath state is still ok but you will get an alarm now for the broken connection regardless if a switch, a line or a controller is broken.

This is much more detailed then counting lines.

History

Why a fork? According to my personal understanding of Nagios, Icinga, etc. are tools for

a) Monitoring and alarming. That means checking values against thresholds (internal or handed over) b) Collecting performance data. These data, collected with the checks, like network traffic, cpu usage or so should be interpretable without a lot of other data.

Athough check_vmware_api.pl is a great plugin it suffers from various things. a) It acts as a monitoring plugin for Nagios etc. b) It acts a more comfortable commandline interfacescript. c) It collects all a lot of historical data to have all informations in one interface.

While a) is ok b) and c) needs to be discussed. b) was necessary when you had only the Windows GUI and working on Linux meant "No interface". this is obsolete now with the new webgui.

c) will be better used by using the webgui because historical data (in most situations) means adjusted data. Most of these collected data is not feasible for alerting but for analysing performance gaps.

So as a conclusion collecting historic performance data collected by a monitored system should not be done using Nagios, pnp4nagios etc.. It should be interpreted with the appropriate admin tools of the relevant system. For vmware it means use the (web)client for this and not Nagios. Same for performance counters not self explaining.

Example: Monitoring swapped memory of a vmware guest system seems to makes sense. But on the second look it doesn't because on Nagios you do not have the surrounding conditions in one view like

- the number of the running guest systems on the vmware server.
- the swap every guest system needs
- the total space allocated for all systems
- swap/memory usage of the hostcheck_vmware_esx.pl
- and a lot more

So monitoring memory of a host makes sense but the same for the guest via vmtools makes only a limited sense.

But this were only a few problems. Furthermore we had

- misleading descriptions
- things monitored for hosts but not for vmware servers
- a lot of absolutely unnesseary performance data (who needs a performane graph for uptime?)
- unnessessary output (CPU usage in Mhz for example)
- and a lot more.

This plugin is old and big and cluttered like the room of my little son. So it was time for some house cleaning. I try to clean up the code of every routine, change things and will try to ease maintenance of the code.

The plugin is not really ready but working. Due to the mass of options the help module needs work.

See history for changes

One last notice for technical issues. For better maintenance (and partly improved runtime) I have decided to modularize the plugin. It makes patching a lot easier. Modules which must be there every time are included with use, the others are include using require at runtime. This ensures that only that part of code is loaded which is needed.

Thanks

- op5 for the original plugin
- Sven Nierlein
- Simon Meggle
- Riccardo Bartels for additional work and creating a "monster" pull request merging a lot of others
- Everyone contributing their time for feedback, patches and issue reports

Contributing

There are many ways to contribute to check_vmware_esx -- whether it be sending patches, testing, reporting bugs, or reviewing and updating the documentation. Every contribution is appreciated!

Releases 2



on Nov 26, 2019

+ 1 release

Contributors 11



Languages

