CHECK_MK BEGINNER GUIDE

Author: Marco Reale

Version: 1.0 – October 2016

Disclaimer:

Please consider this guide just as a bunch of notes and NOT as a professionally written document. My intention is to give something back to the community and I haven't any relation with the company behind Check_MK.
I assume no responsibility for the accuracy, completeness, or usefulness of any information or for damages resulting from the procedures provided. Furthermore, this documentation is supplied "as is" without guarantee or warranty, expressed or implied, including without limitation, any warranty of fitness for a specific purpose.
I sincerely thank the Check_MK mailing list users because without their help I would not have been able to write this guide.

Special thanks to:

Jolyon Brown: Help in translation
Mike Hulsman: Solution for Monitoring Microsoft Terminal Services
Brian Binder: Solution for Monitoring Microsoft Windows Event Log
Andreas Döhler: Explanation about Folders,Tags & Hostgroup
Evy Bongers: Explanation about Folders,Tags & Hostgroup

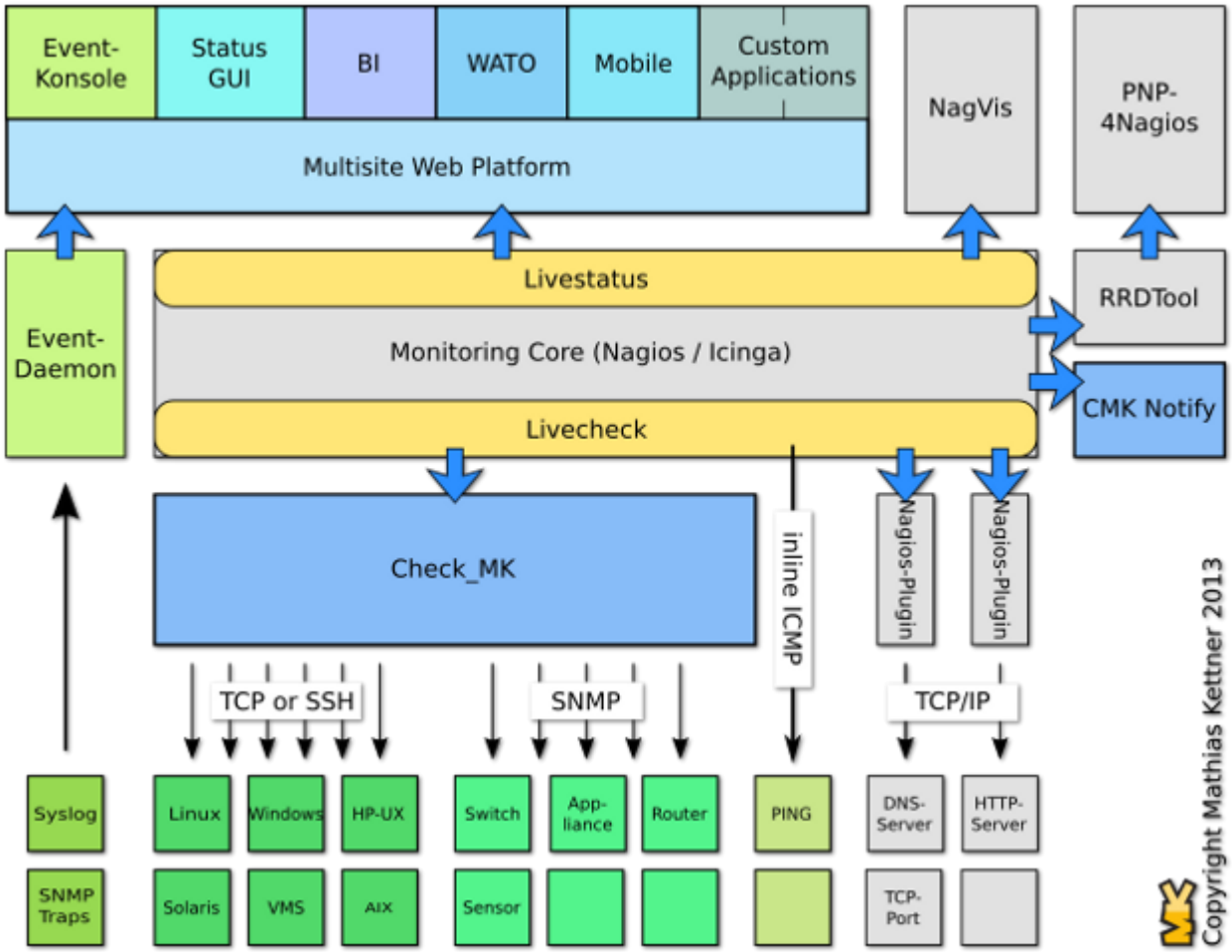
Apologies to anyone I've missed.

Summary

INTRODUCTION

Every system administrator should know the current state of infrastructure they are responsible for. There is nothing worse than realising much too late that a service is down or, even worse, to have users notify you of problems you hadn't yet noticed.
A good monitoring solution provides automated reporting of errors and malfunctions allowing immediate intervention. In addition, this automation frees IT personnel from having to keep constant watch over all infrastructure - servers, desktop computers, applications, traffic, etc. so they can use their time for other tasks.
Unfortunately, not all companies understand the importance of such monitoring until there are serious problems that affect their business. In my career I've had situations where managers asked me for the reasons behind serious and continuous performance problems - and why we were not able to quickly identify them. I've always replied that without a good monitoring solution, we were blind. I don't want to claim that monitoring itself prevents any kind of outage or can ensure 100% uptime, because clearly there are other important factors to consider (even organizational aspects). But believe me, it is extremely important and helps prevent many potential outages.
In my opinion, an Enterprise monitoring solution should provide the following features out of the box: scalability, multi-tenancy, granular access to hosts and services, customizable dashboards, notifications, good performance graphs, automatic inventory of services being monitored, certified plugins for all enterprise class hardware/software, understanding of parent & child relationships, flexibility in creating custom checks and, last but not least, should be easy to install, use and maintain.
The market has plenty of solutions (both free and commercial), but most of them are difficult and time-consuming with a steep learning curve and/or quite expensive. Over the last few years I've worked with many products and, even though I must admit that I had some nice results, I never really found something that completely satisfied me. I always find them lacking something or other.
One day though I came across Check_MK. A solution that, in a nutshell, claimed to make Nagios much easier and more powerful to use.
As the official site states (http://mathias-kettner.com/check_mk.html), Check_MK is a comprehensive IT monitoring solution in the tradition of Nagios. The main developer for the product is Mathias Kettner and the company he has formed around it is located in Munich, Germany.

The following diagram (taken from the official website) shows how with the help of Check_MK and Nagios, a complete monitoring solution can be assembled. The coloured boxes represent the components of the Check_MK-Project.



Check_MK is available as a 100% open source package (known as the "Raw Edition (CRE)") and as a professionally supported "Enterprise Edition (CEE)" that comes with a lot of additional features such as:

-Agent bakery (packaging of individual monitoring agents)
-High performance and low latency via Check_MK Micro Core
-Reporting: Creation of individualized reports in PDF format
-Improved performance on distributed setup
-Auto expiration of acknowledged alarms
-Recurring planned-downtimes

You also can deploy the CEE by using or a hardware or software appliance.
For a complete comparison, please have a look at http://mathias-kettner.com/check_mk_introduction.html

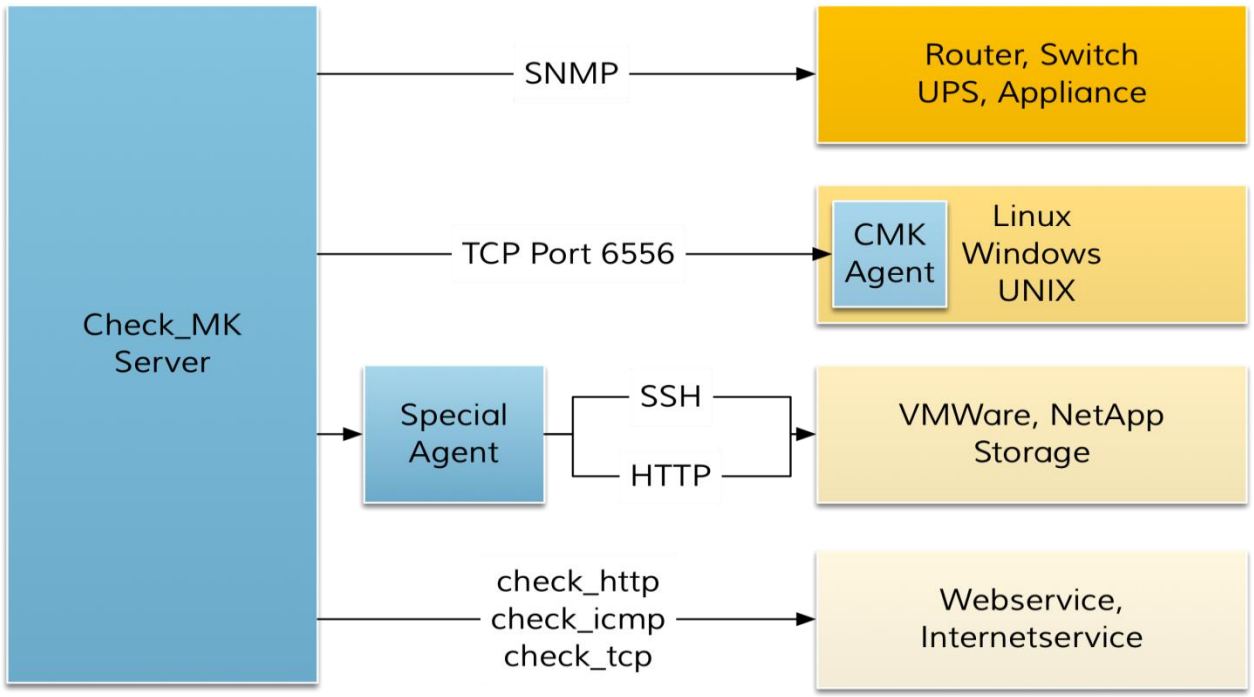
In my opinion, these are the main Check_MK key features:

-Fully compatible with Nagios
-Excellent performance even on large environments
-Scale-out/Distributed monitoring with centralized management
-Graphical User Interface (WATO)
-Shorter learning curve compared to other Nagios solutions

-Hundreds of certified plugins and supported devices
-Agent based monitoring for OSs and SNMP for network devices

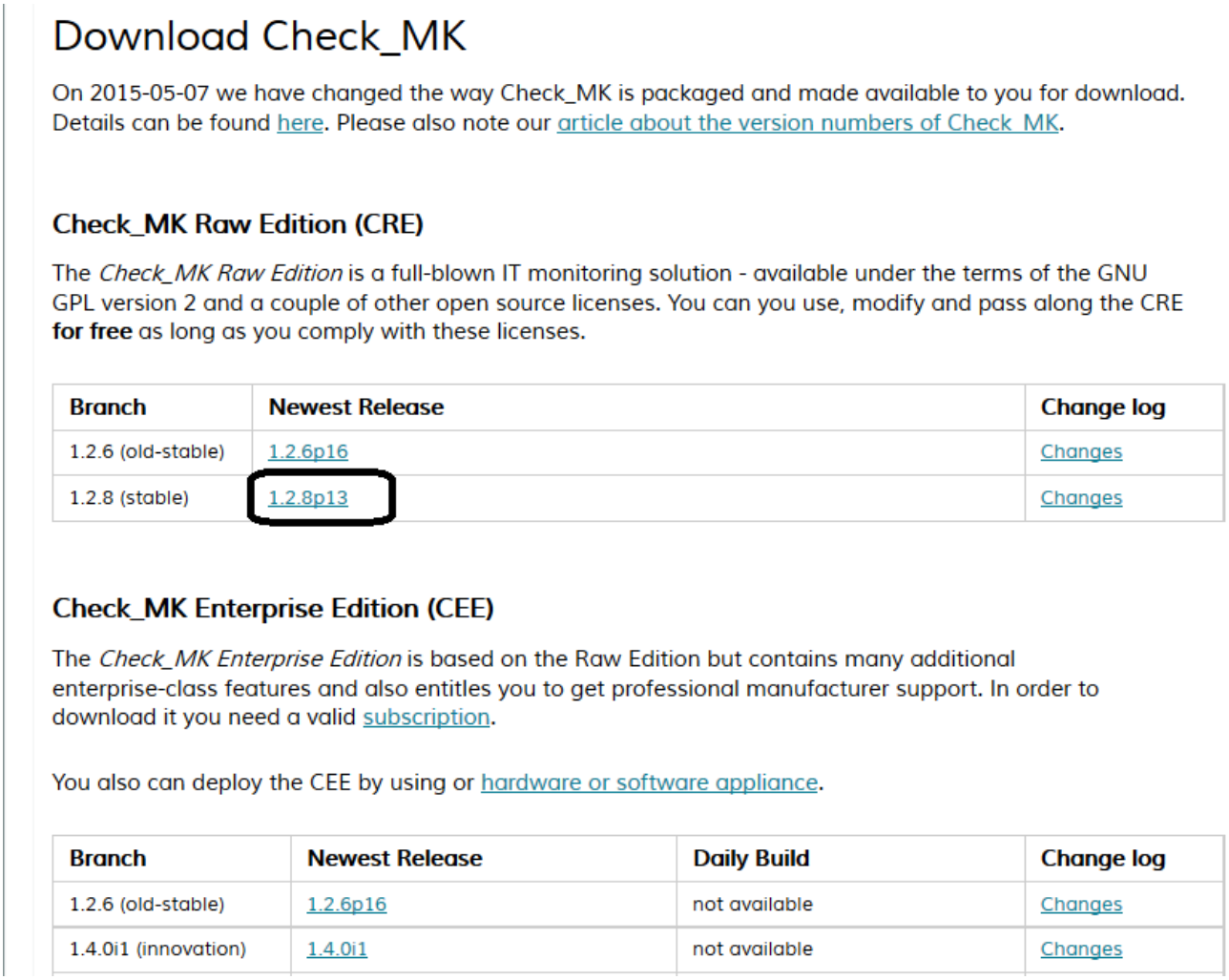
The following table shows the four different ways that Check_MK can access services to be monitored:

(ref: https://mathias-kettner.com/cms_wato_monitoringagents.html)



Check_MK Setup

In this guide I'm going to show how to set up and get started with Check_MK Raw Edition on a Centos 7.2 virtual machine. I always suggest installing the latest stable version that, at the time of writing (October 2016), is 1.2.8p13.



Download Check_MK

On 2015-05-07 we have changed the way Check_MK is packaged and made available to you for download. Details can be found here. Please also note our article about the version numbers of Check_MK.

Check_MK Raw Edition (CRE)

The *Check_MK Raw Edition* is a full-blown IT monitoring solution - available under the terms of the GNU GPL version 2 and a couple of other open source licenses. You can you use, modify and pass along the CRE **for free** as long as you comply with these licenses.

Branch	Newest Release	Change log
1.2.6 (old-stable)	1.2.6p16	Changes
1.2.8 (stable)	1.2.8p13	Changes

Check_MK Enterprise Edition (CEE)

The *Check_MK Enterprise Edition* is based on the Raw Edition but contains many additional enterprise-class features and also entitles you to get professional manufacturer support. In order to download it you need a valid subscription.

You also can deploy the CEE by using or hardware or software appliance.

Branch	Newest Release	Daily Build	Change log
1.2.6 (old-stable)	1.2.6p16	not available	Changes
1.4.0i1 (innovation)	1.4.0i1	not available	Changes

Requirements for TEST environment:

-Centos 7 64bit with 2vCPU, 4GB RAM, 30GB HD
-Working internet connection
-EPEL repository enabled
-SELinux disabled or properly configured

Please note that the above list is just for a TEST environment; to properly size a production server, there are many variables to consider such as the number of monitored services and the hardware you are going to place Check_MK on (carefully distinguishing between physical and virtual hardware).
There are some handy notes about sizing considerations at the following link: https://mathias-kettner.de/checkmk_checkmk_benchmarks.html

Step by step setup:

1) Install Centos 7.2 64 bit

2) Check internet connection and enable EPEL repository

3) Download the last version of Check_MK and place it in */tmp/setup_checkmk/*
(Please note that in this guide I started with 1.2.8p11 - just because later I'll show how to update to 1.2.8p13. If this is the first time you are going to setup Check_mk, download the latest version!)

4) *cd /tmp/setup_checkmk/*

5) *yum localinstall -ivh check-mk-raw-1.2.8p11-el7-36.x86_64.rpm*

6) Create your first OMD site. You just have to choose a site name, like prod or test or whatever you like (in this example I have chosen "*mysite*"). Then, as root user, you simply type:

   omd create mysite

7) Using a browser, point to http://ip/mysite

8) Login to using default credentials:

   username: omdadmin
   password: omd

That's all! As you can see the setup is really easy. I'd even describe it as being "windows like" - but without the need to reboot ☺

WATO – The Graphical User Interface

WATO is a nice and powerful GUI through which it is possible to manage hosts and services being monitored with Check_MK. Just please note that by using WATO, you can avoid the use of the command line for many tasks but not all of them. Why? The best explanation is provided by Mathias on his website from which I took the following screenshot:

(ref: https://mathias-kettner.de/checkmk_wato.html)

WATO - Check_MK's Web Administration Tool

Dieser Artikel wird nicht mehr gepflegt und ist unter Umständen nicht mehr gültig!

1. Introduction

WATO is Check_MK's new graphical administration tool. It is a web based user interface for managing hosts and services to be monitored with Check_MK. However, WATO is no tool for configuring all aspects of Check_MK and Nagios. Why?

The basic idea behind WATO is that - when it comes to monitoring - usually one or few persons are responsible for setting up and maintaining the actual monitoring server. They spend a lot of time with the system and its internals and usually have no difficulties with editing text based configuration files - or even prefer them over a GUI.

The "users" of the monitoring, however, just need their systems to be monitored but do not have the time to learn how to write valid configuration files. Nor is it their job. So whenever they need any modification in the monitoring - for example if a new server has been set up, a switch configuration has changed or a database instance has been removed - they ask the monitoring team to adapt their configuration accordingly. Those changes make up a substantial part of the daily workload of the monitoring team.

WATO allows you to move these daily tasks to the users by providing them a GUI for managing their hosts and services themselves. The monitoring team can spend their time with their actual work - tuning the system, implementing new checks, configuring general rules, and so on.

1.1. Why not using NagiosQL, LConf, NConf, Centreon or other GUIs?

The first simple reason is: They do not support Check_MK and probably never will. Another reason is: WATO directly supports Check_MK's inventory mechanism and thus an auto-detection of services. And last but not least, WATO takes into account that the different needs of monitorings admins and users should be reflected in the GUI.

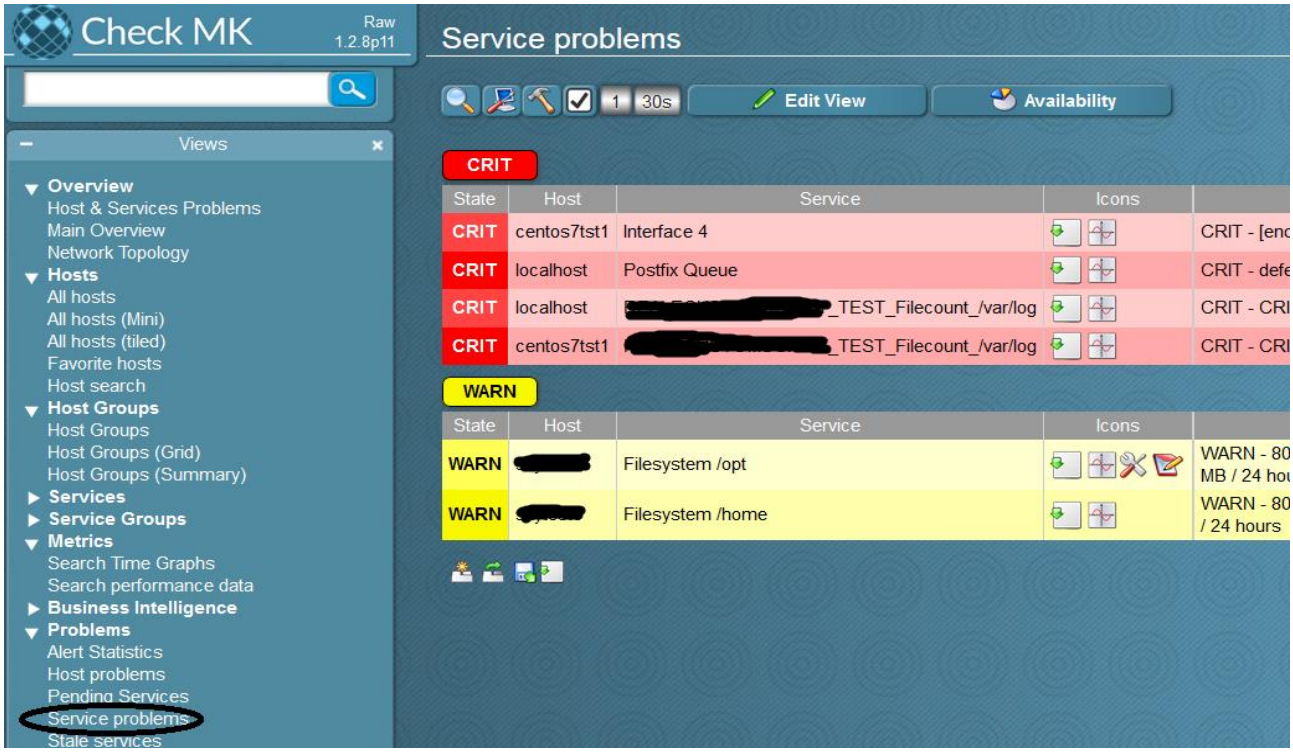
That said, I found that I could do most tasks using just the GUI. Moreover every new version seems to add some WATO module so which brings into the GUI some tasks which previously had to be performed manually.

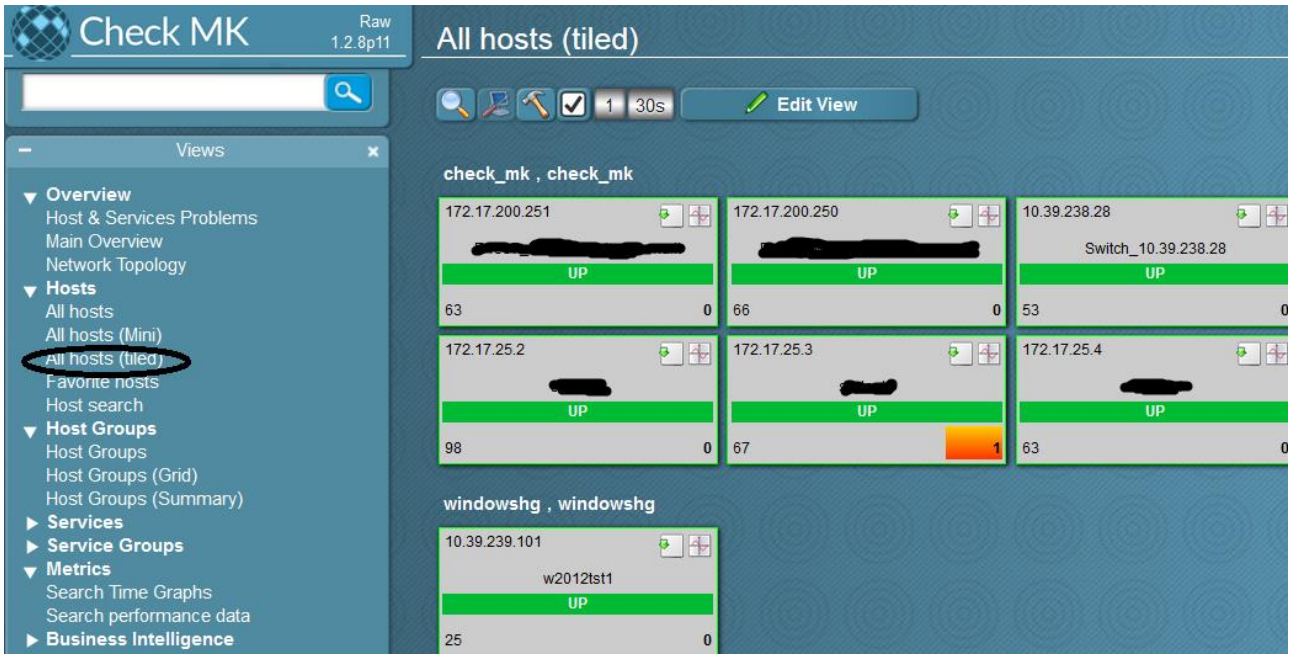
This is the main WATO window that provides a global overview of Host and Services statistics as well as a list of recent events.

On the left side there are two main sections: *Views* and *Configuration*

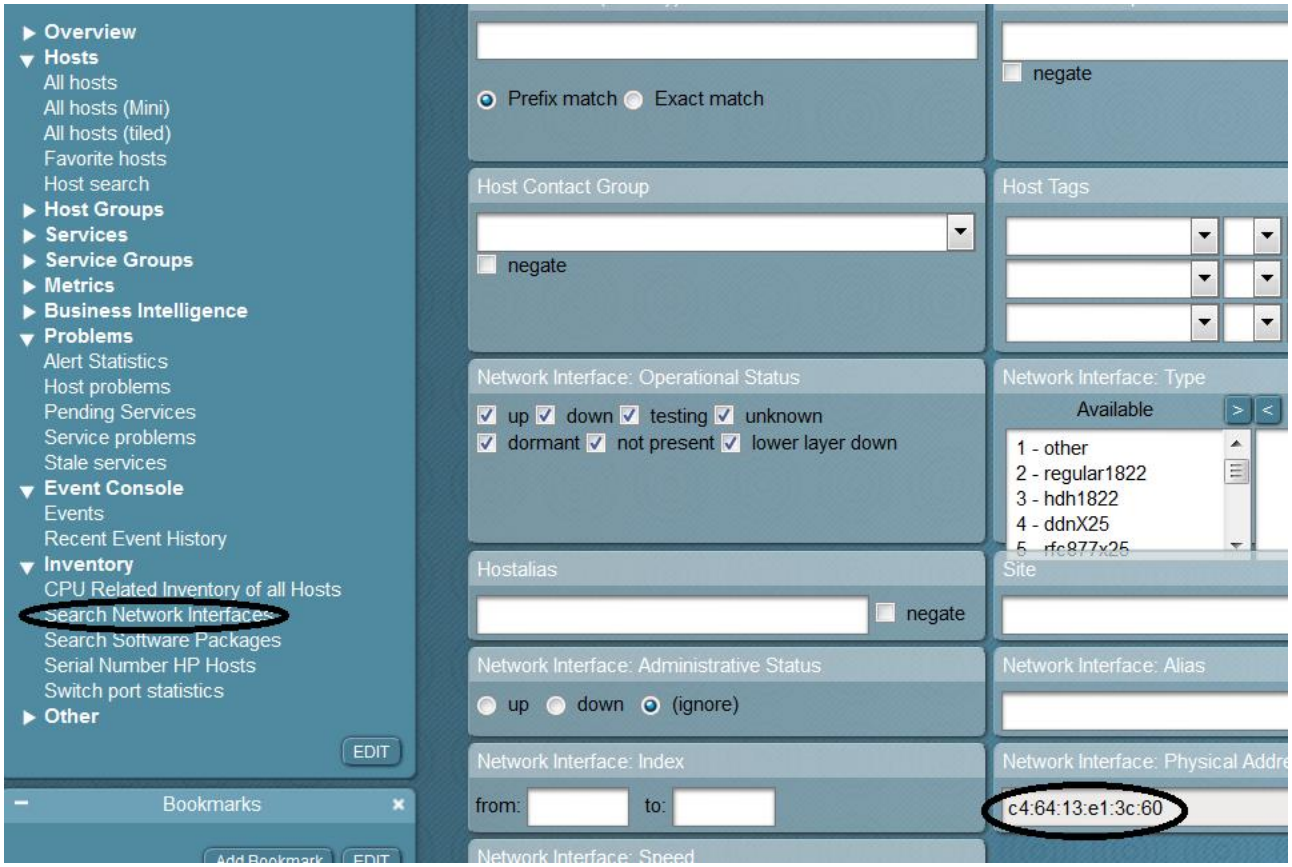
Views - Pane

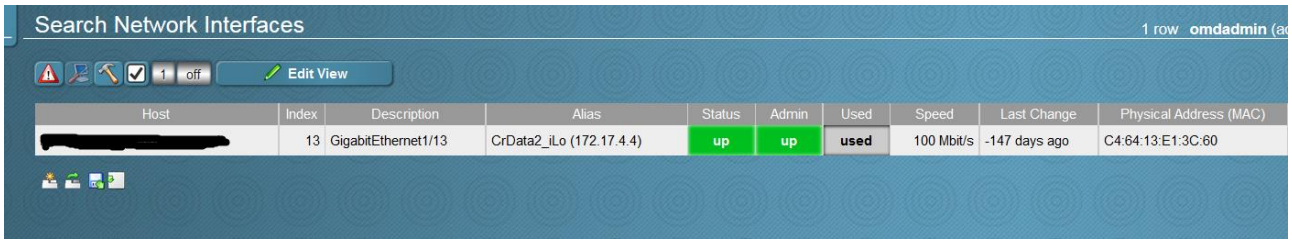
In this section there are many views of different components like these:





It's also possible do some useful searches. For example - did you ever try to find the switch port of a specific MAC or IP address? With WATO, this can be done with just a couple of clicks.





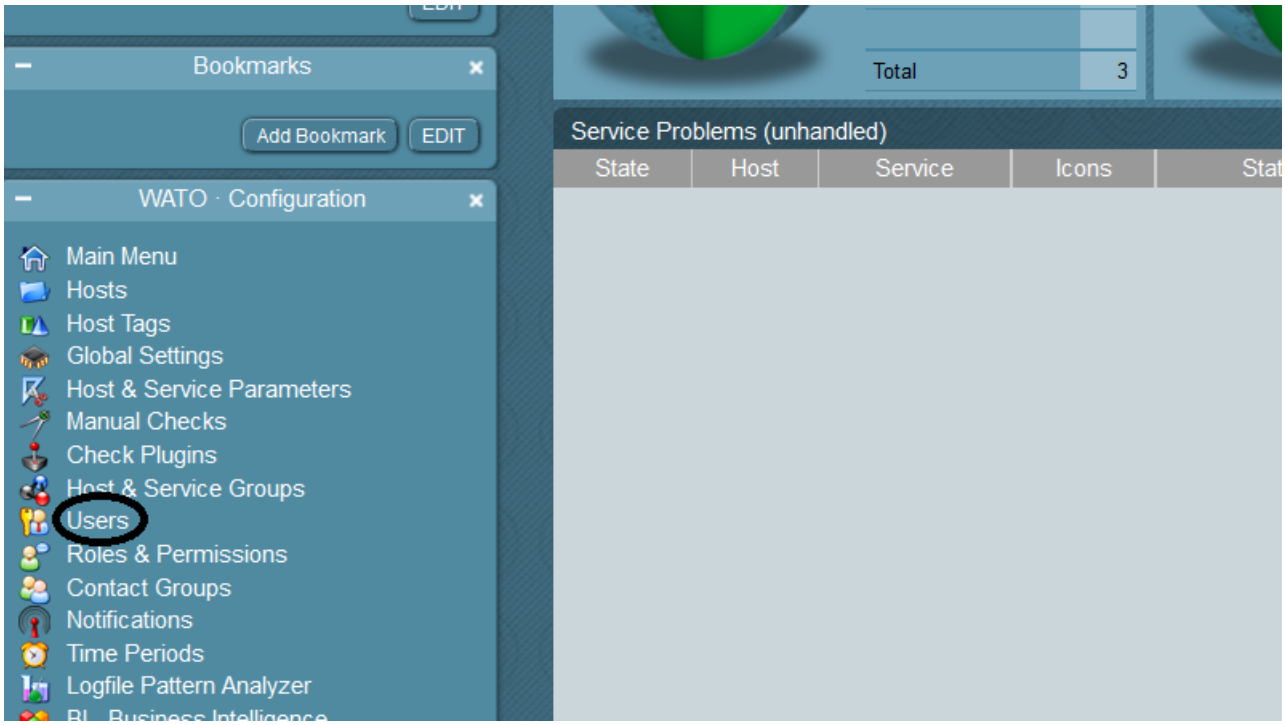
Configuration – Pane

This menu is divided into many sections but by clicking on *Main Menu* you can access all of them from a single point

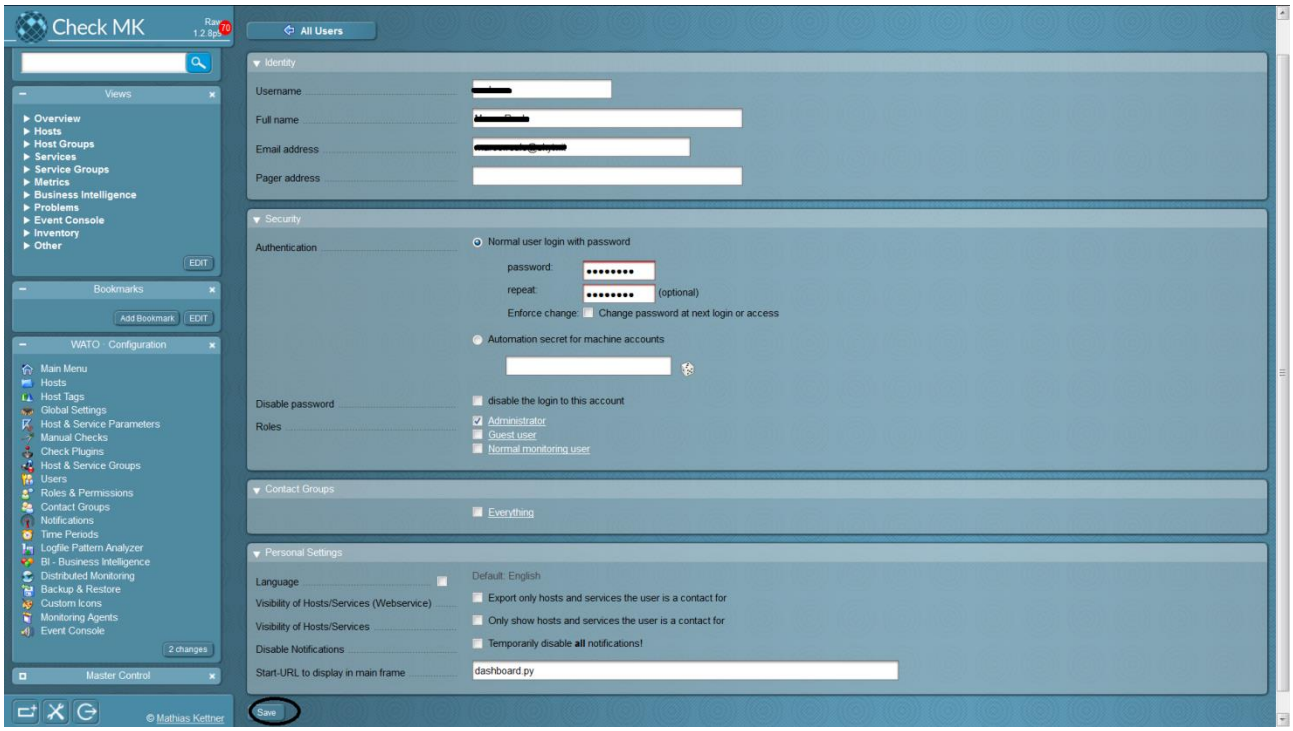
I'm not going to describe each sub-menu but will cover a few of them in the following section.

Users

One of the first tasks that should be performed after the setup is the creation of users. Everyone who is going to use check_MK should have their own custom credentials. This is done using WATO: *Users*

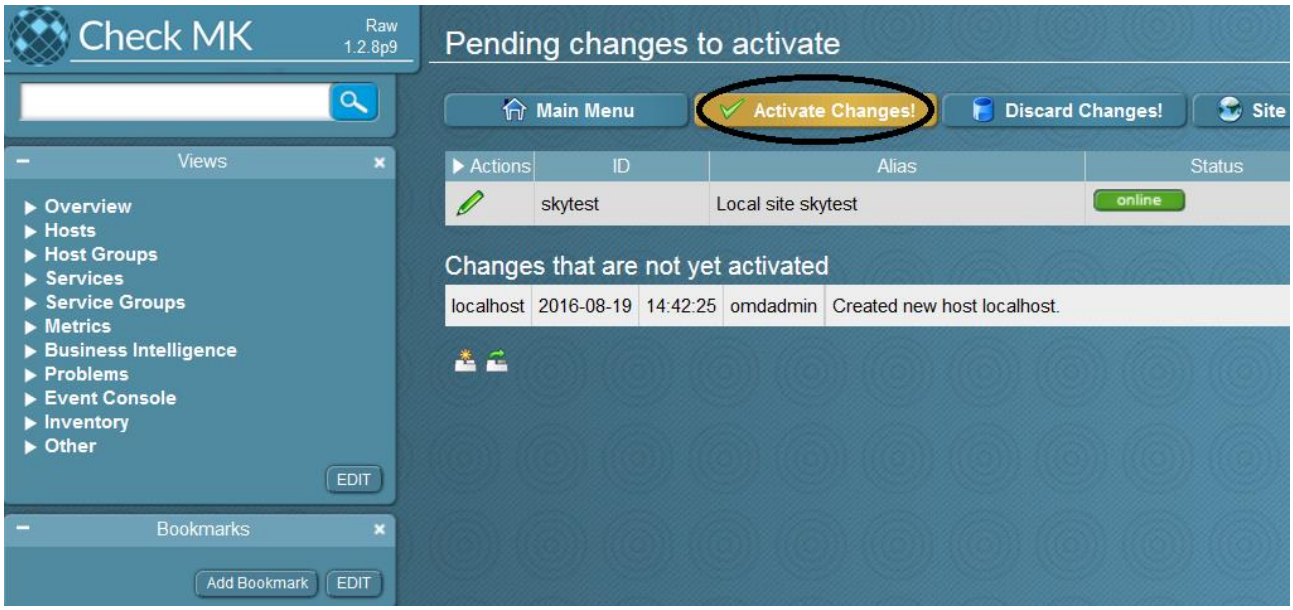


It's possible to create a new user by cloning an existing one:


Check MK
Raw
1.2.8p
All Users
Views
Overview
Hosts
Host Groups
Services
Service Groups
Metrics
Business Intelligence
Problems
Event Console
Inventory
Other
EDIT
Bookmarks
Add Bookmark
EDIT
WATO - Configuration
Main Menu
Hosts
Host Tags
Global Settings
Host & Service Parameters
Manual Checks
Check Plugins
Host & Service Groups
Users
Roles & Permissions
Contact Groups
Notifications
Time Periods
Logfile Pattern Analyzer
BI - Business Intelligence
Distributed Monitoring
Backup & Restore
Custom Icons
Monitoring Agents
Event Console
2 changes
Master Control
© Mathias Kettner
Identity
Username
Full name
Email address
Pager address
Security
Authentication
Normal user login with password
password:
repeat:
(optional)
Enforce change:
Change password at next login or access
Automation secret for machine accounts
Disable password
disable the login to this account
Roles
Administrator
Guest user
Normal monitoring user
Contact Groups
Everything
Personal Settings
Language
Default: English
Visibility of Hosts/Services (Webservice)
Export only hosts and services the user is a contact for
Visibility of Hosts/Services
Only show hosts and services the user is a contact for
Disable Notifications
Temporarily disable all notifications!
Start-URL to display in main frame
dashboard.py
Save

Apply Changes

Whenever changes are made in the configuration, we need to restart check_mk by clicking on the *Changes* button followed by *Activate Changes*

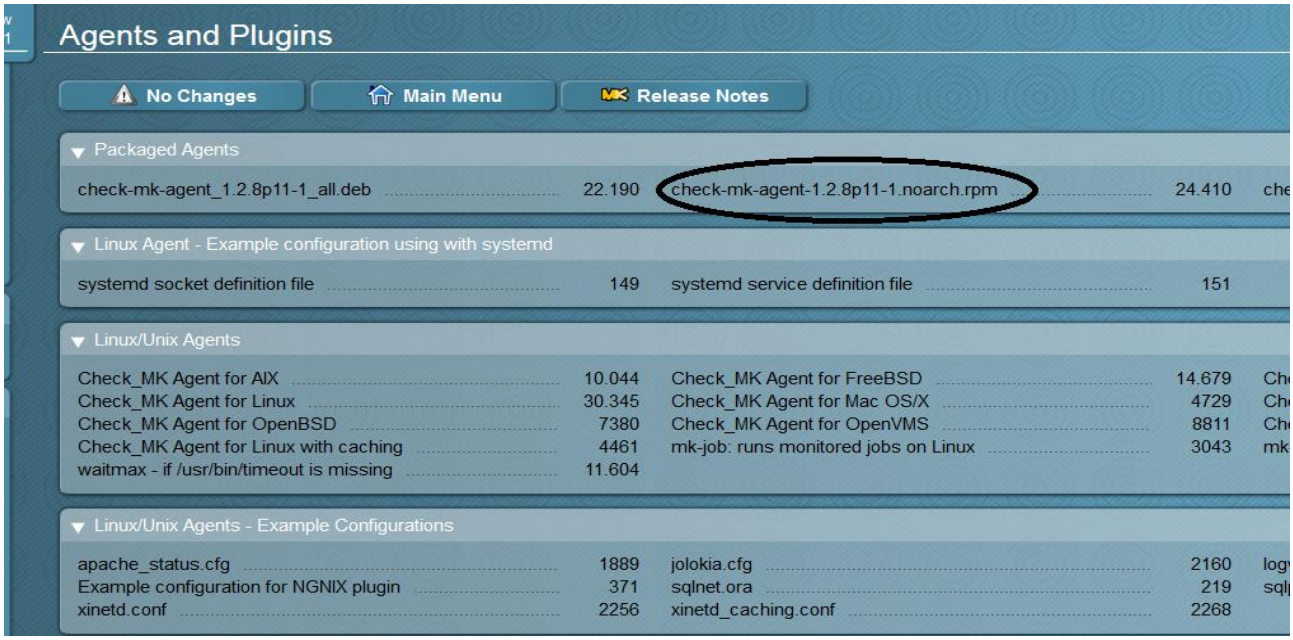


Managing agents

Agents for many operating systems are available in *WATO, Monitoring Agents*. There are *rpm* and *deb* packages but a manual installation is possible too. The Enterprise version provides a feature called *agent bakery* that allows the creation of custom packages; combined with the *Automatic Agent Update feature* available since version 1.2.8, the effort needed to update agents is extremely reduced, especially in large environments. Running agents will listen on port TCP 6556.

Agent Installation on Linux

We are going to install the agent on localhost (where check_mk is running) using *rpm.* Installing the Agent via RPM or DEB is very easy. All you have to do is to make sure `xinetd` is installed first and then install the package.
Click on *WATO, Monitoring Agents* and select check_mk-agent rpm



You can download or copy it manually:

```
[root@checkmktst1 linux]# pwd
/tmp/setup_checkmk/agents/linux
[root@checkmktst1 linux]# wget http://localhost/mysite/check_mk/agents/check-mk-agent-1.2.8p11-1.noarch.rpm
--2016-08-19 14:37:13--  http://localhost/mysite/check_mk/agents/check-mk-agent-1.2.8p11-1.noarch.rpm
Resolving localhost (localhost)... ::1, 127.0.0.1
Connecting to localhost (localhost)|::1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 24407 (24K) [application/x-rpm]
Saving to: 'check-mk-agent-1.2.8p11-1.noarch.rpm.1'

100%[======================================================================>] 24,407      --.-K/s   in 0s

2016-08-19 14:37:13 (473 MB/s) - 'check-mk-agent-1.2.8p11-1.noarch.rpm.1' saved [24407/24407]

[root@checkmktst1 linux]# rpm -ivh check-mk-agent-1.2.8p11-1.noarch.rpm
Preparing...                ########################################### [100%]
Updating / installing...
   1:check-mk-agent-1.2.8p11-1        ########################################### [100%]
Reloading xinetd...
Redirecting to /bin/systemctl reload  xinetd.service
```

The xinetd file should look like this:

```
/etc/xinetd.d/check_mk
service check_mk
{
        type            = UNLISTED
        port            = 6556
        socket_type     = stream
        protocol = tcp
        wait            = no
        user            = root
        server          = /usr/bin/check_mk_agent
# configure the IP address(es) of your Nagios server here:
#       only_from       = 127.0.0.1 10.0.20.1 10.0.20.2
        disable         = no
}
```

Agent Installation on Windows

Download "check_mk_agent.msi" and install it on all servers that you need to monitor.

Restart the agent using:

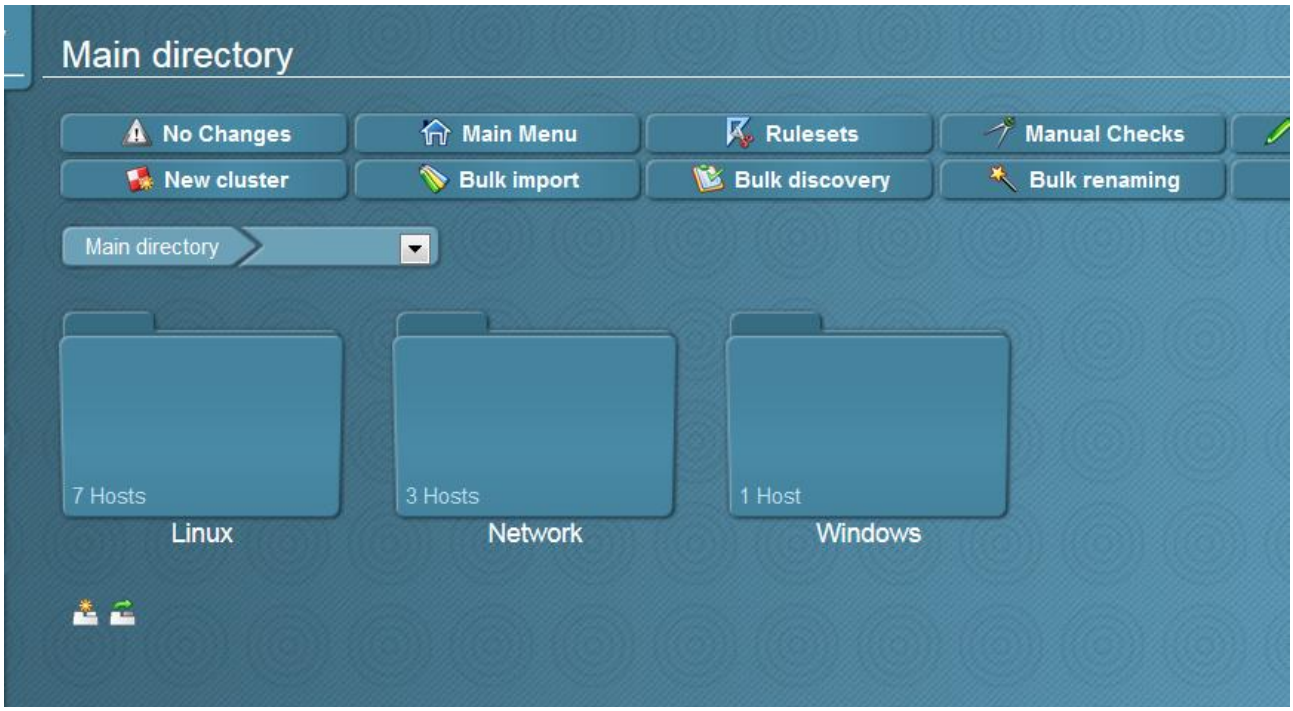
net stop check_mk_agent && net start check_mk_agent

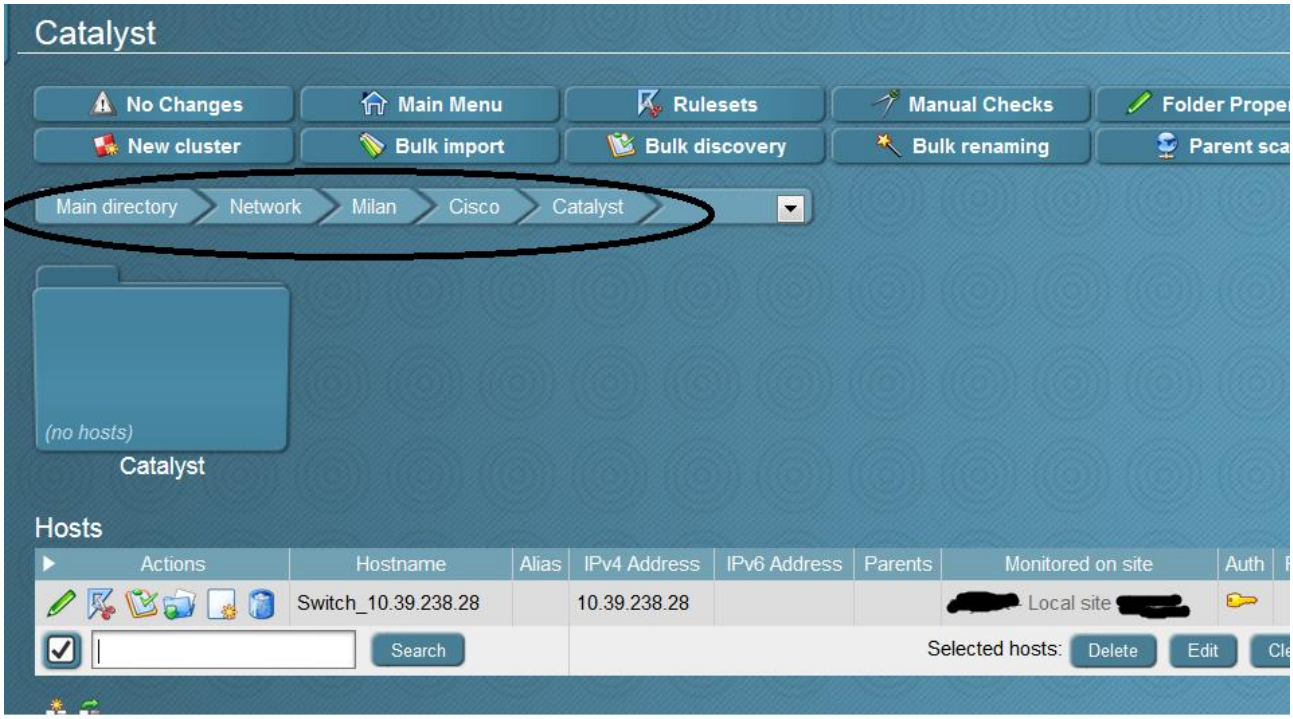
Devices Management

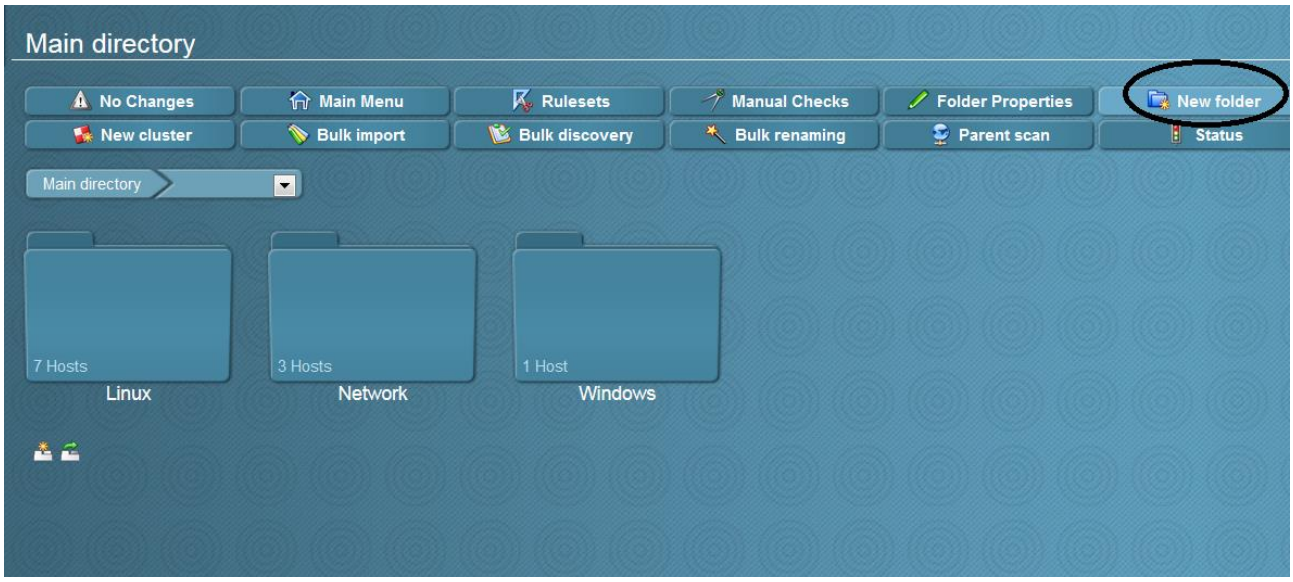
Managing devices doesn't just mean adding or removing devices but also applying checks, creating rules, thresholds and (last but not least) organizing them. In check_MK, this is achieved using *Folders, Tags and Hostgroup.*
Basically, these are just different ways to achieve a common purpose: organizing devices so that configuring them is easy even with a large number of hosts.
Managing hundreds or even thousands or devices could be very difficult without a proper classification that allows rules to be applied to groups of objects instead of single entities.
The best analogy that comes to my mind is Microsoft Active Directory that allows policies to be created for the entire domain (the root), sites or even just organizational units.
The question is, which one of them should be used?
Well, I opened a thread about this topic on the Check_MK English mailing list and I received some good advice from expert users which, most of the time, suggest using both of them.
Let's say you have 1000 devices in one site; in this case you can create folders for each category such as Windows servers, Linux, UPS, Storage etc.
If you have 20 sites and 500 hosts you may want to create one folder for each site e.g. London, New York, Paris and so on and then create subfolders for Linux, Windows, UPS etc.
But another option would be to create host tags.
Also hostgroups can make searching for hosts easier. You can create directories per project and assign host tags accordingly. Using these host tags, you can assign hosts to project specific hostgroups, which makes it possible to search for all hosts in a specific project. Also, you can allow customers or users within your company access to their specific projects by making them contacts for their project specific host groups.

Folders

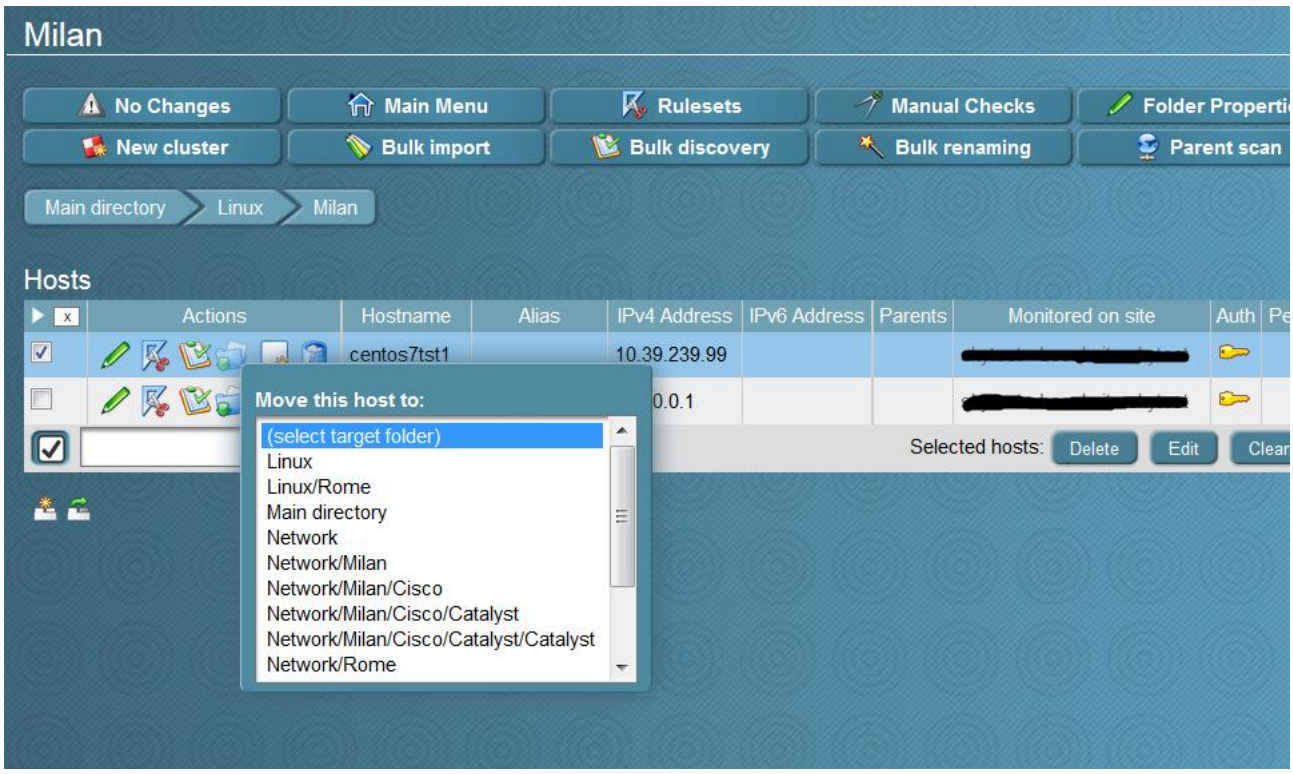
By default there is just the *Main directory* where devices are placed if no specific folder is chosen for them.
Click on *WATO, Hosts*, *New folder* to add more folders.
In this picture there are some folders within the Main directory and each one of them contains other sub folders and devices accordingly to their topology


Main directory
No Changes
Main Menu
Rulesets
Manual Checks
New cluster
Bulk import
Bulk discovery
Bulk renaming
Main directory
7 Hosts
Linux
3 Hosts
Network
1 Host
Windows


Catalyst
No Changes
Main Menu
Rulesets
Manual Checks
New cluster
Bulk import
Bulk discovery
Bulk renaming
Main directory
Network
Milan
Cisco
Catalyst
(no hosts)
Catalyst
Hosts
Actions
Hostname
Alias
IPv4 Address
IPv6 Address
Parents
Monitored on site
Auth
Switch_10.39.238.28
10.39.238.28
Local site
Search
Selected hosts:
Delete
Edit

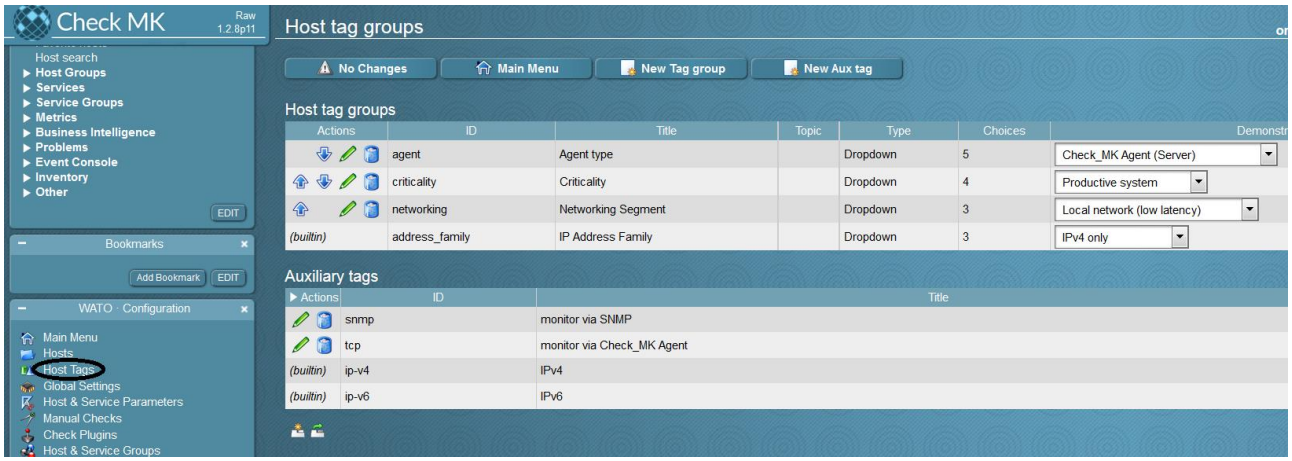


To place a device in folders, select the device and click on the *folder* icon

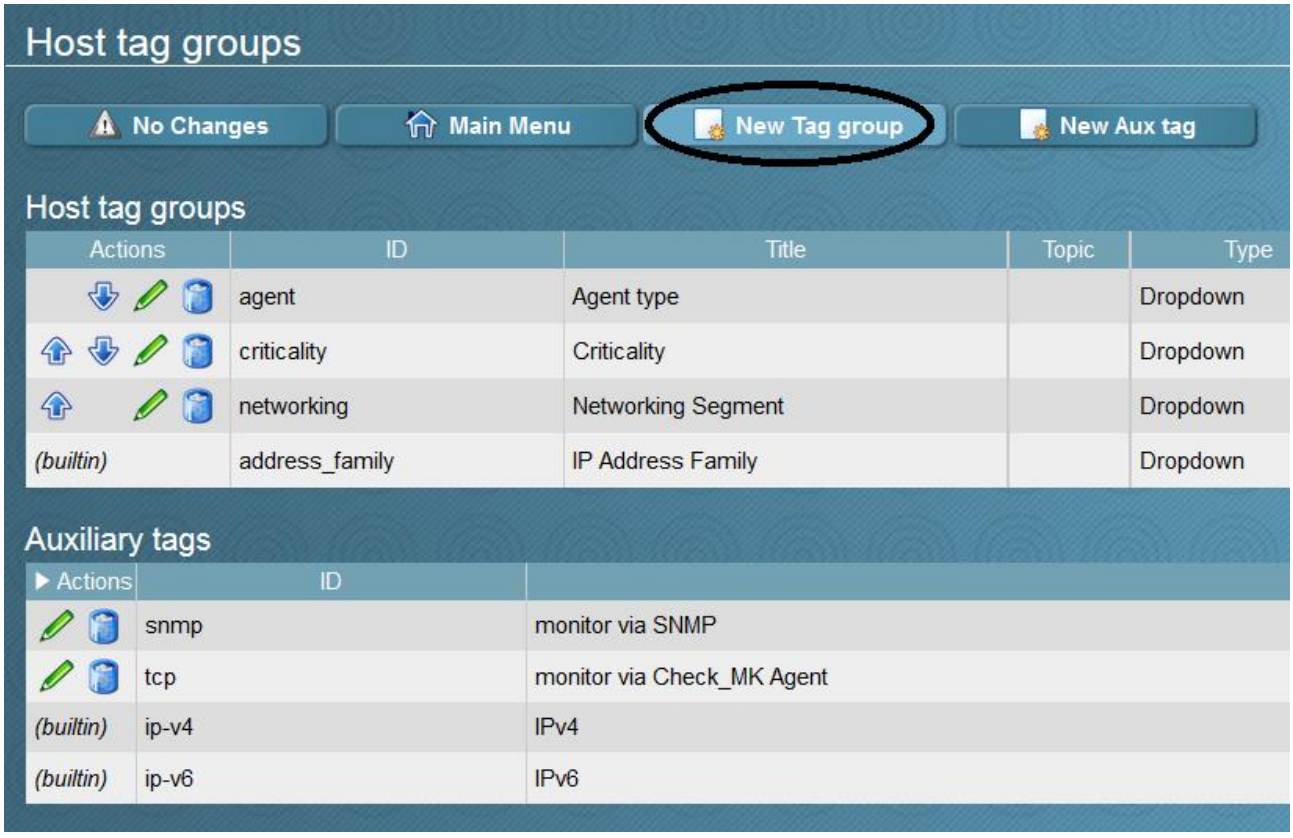


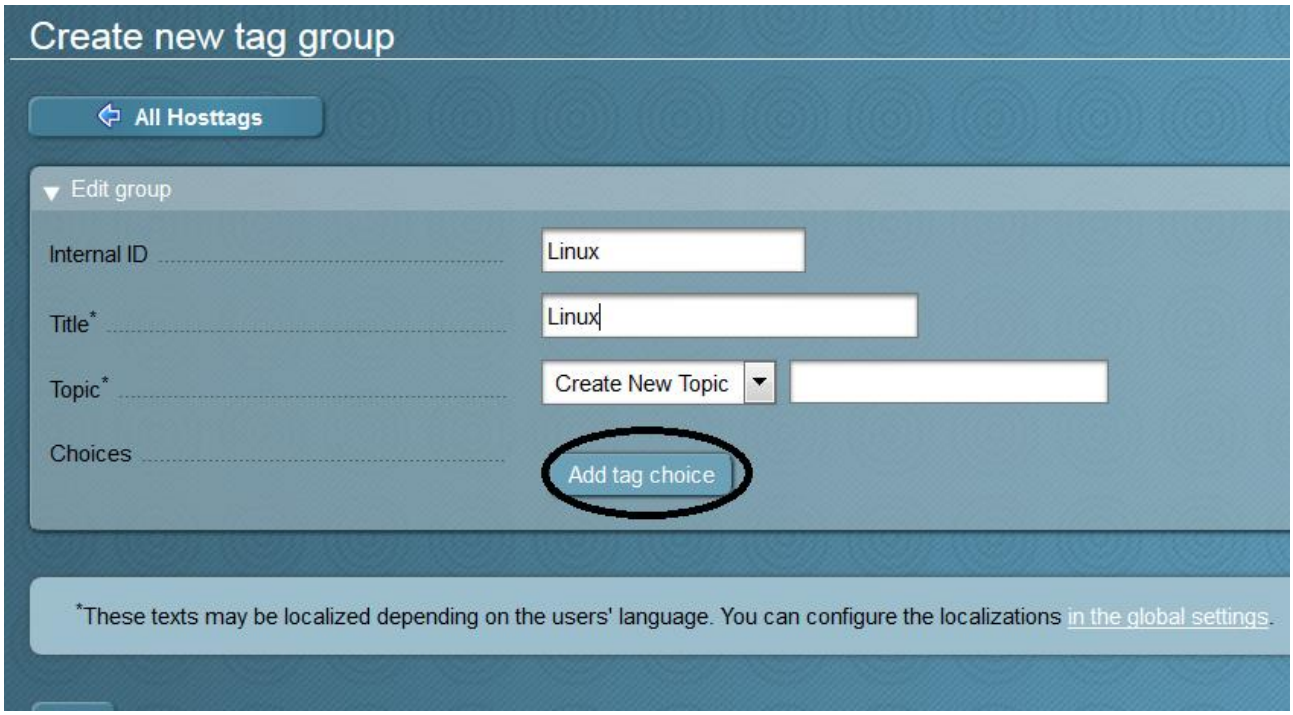
Tags

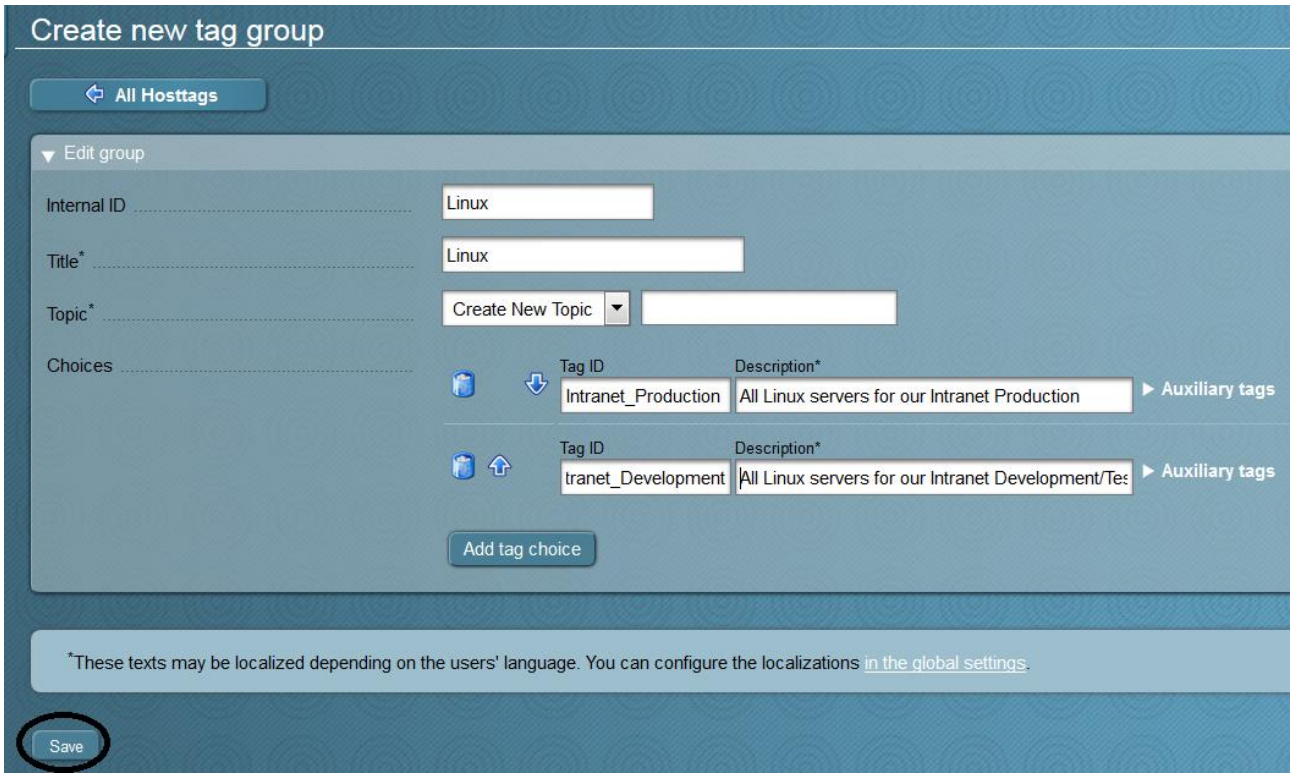
Clicking on *WATO, Host Tags* shows that there are some *Tags* already in place



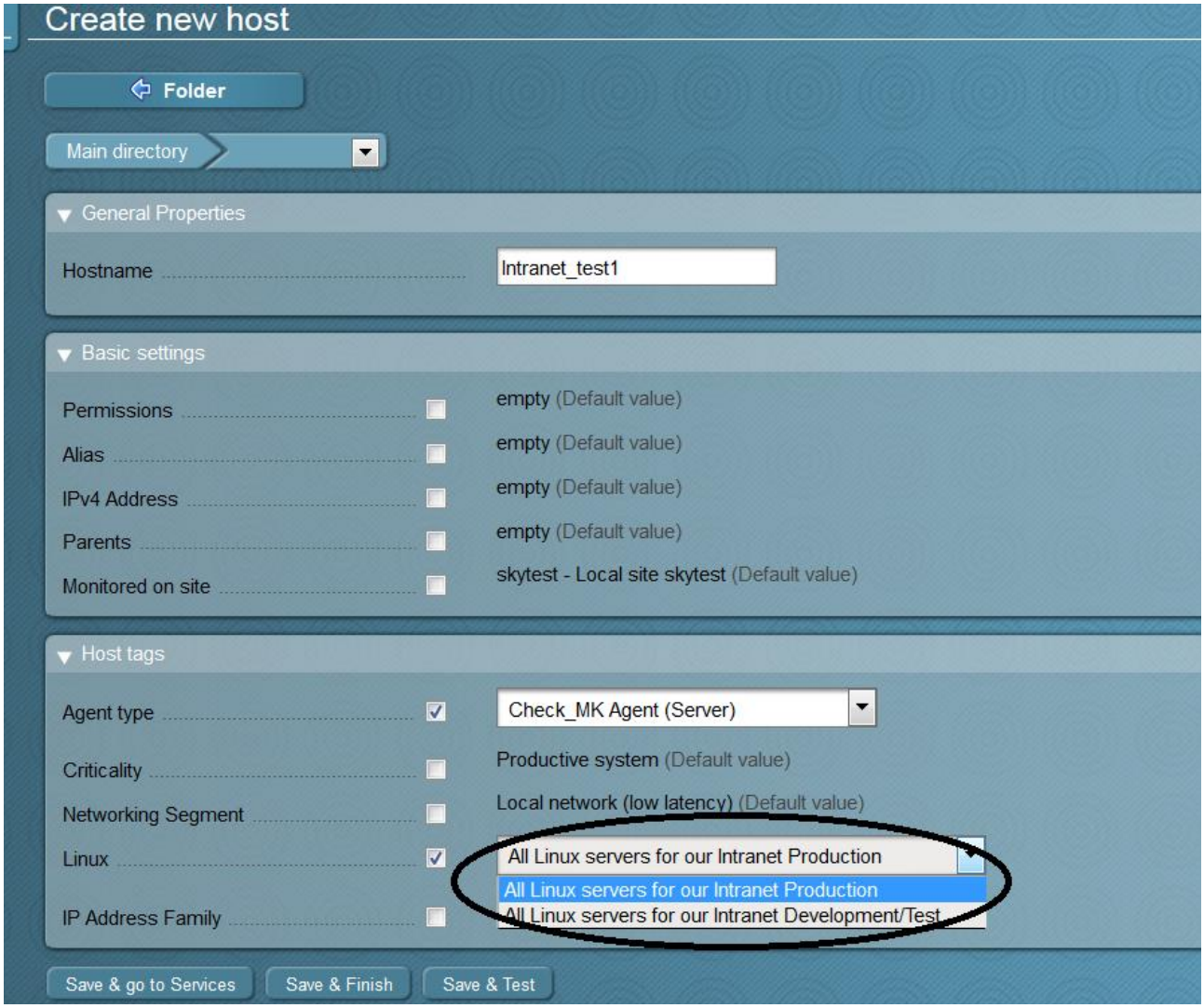
To create a new Tag:




Create new tag group
All Hosttags
Edit group
Internal ID
Linux
Title*
Linux
Topic*
Create New Topic
Choices
Add tag choice
*These texts may be localized depending on the users' language. You can configure the localizations in the global settings.

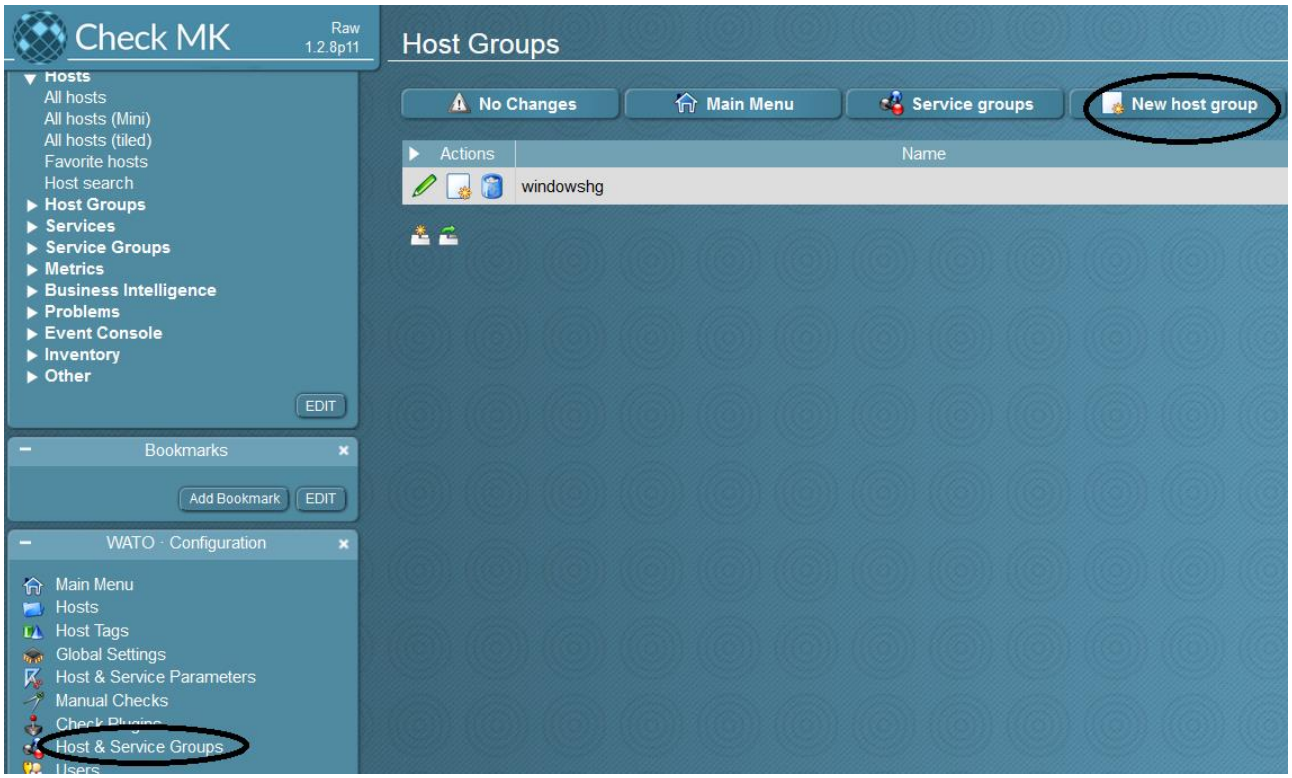

Create new tag group
All Hosttags
Edit group
Internal ID
Linux
Title*
Linux
Topic*
Create New Topic
Choices
Tag ID
Description*
Intranet_Production
All Linux servers for our Intranet Production
Auxiliary tags
Tag ID
Description*
tranet_Development
All Linux servers for our Intranet Development/Tes
Auxiliary tags
Add tag choice
*These texts may be localized depending on the users' language. You can configure the localizations in the global settings.
Save

Tags can be applied during the *New host* wizard process - or after by editing the properties of the device.



Hostgroup

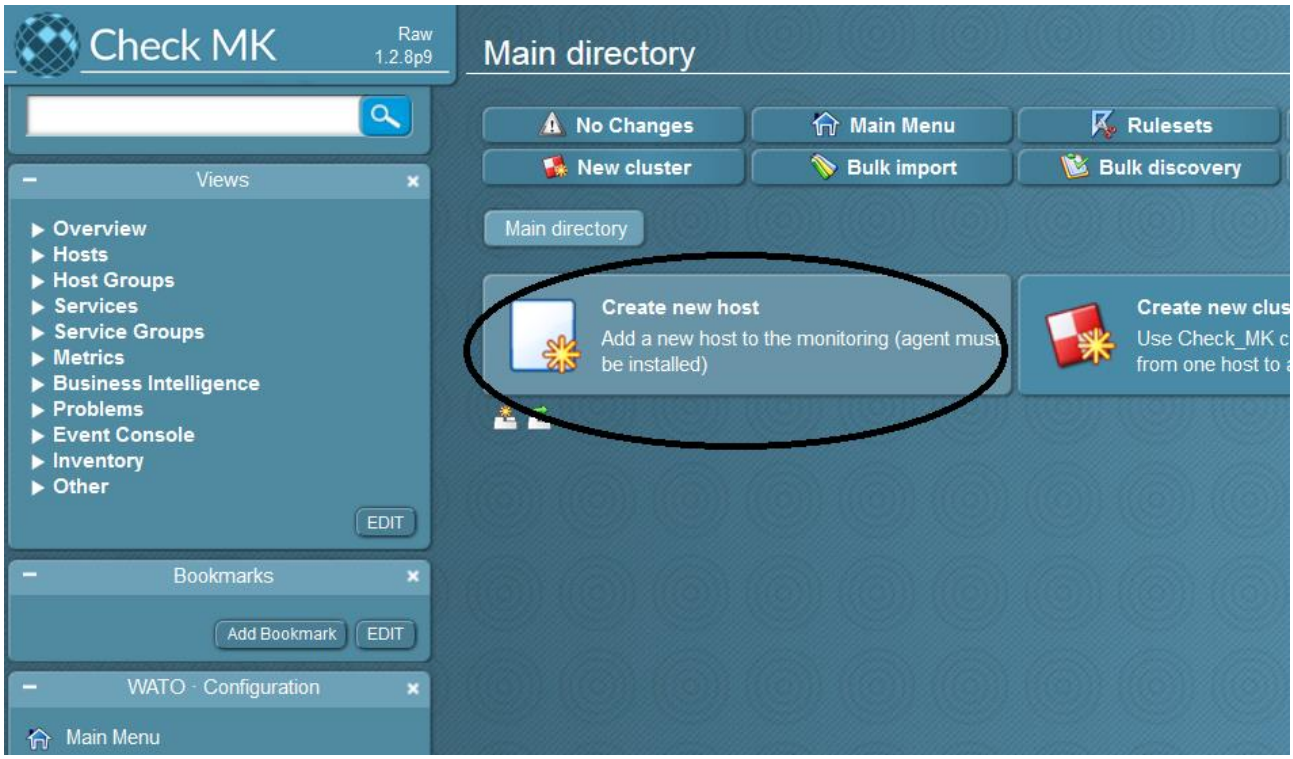
To create a new hostgroup click on *WATO, New host group*



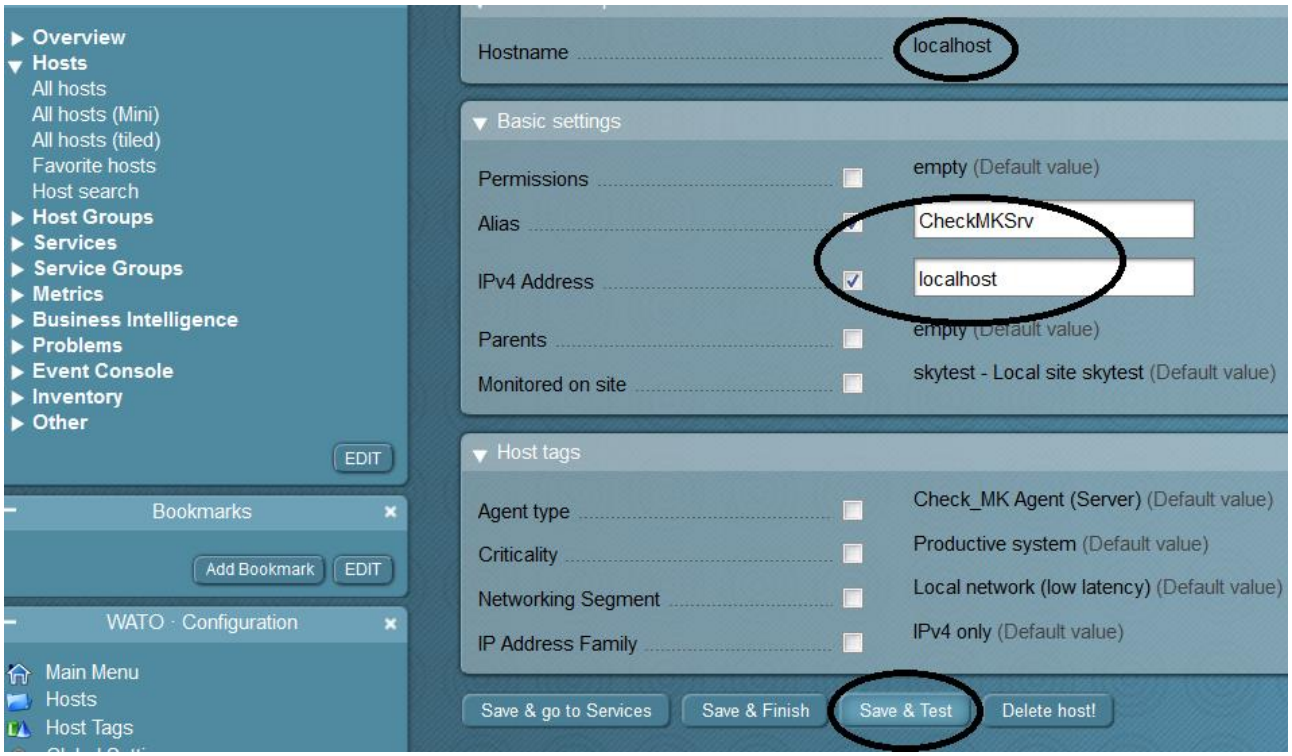
To do Host group assignment: *WATO, Host & Service Parameters, Grouping*

Linux Devices

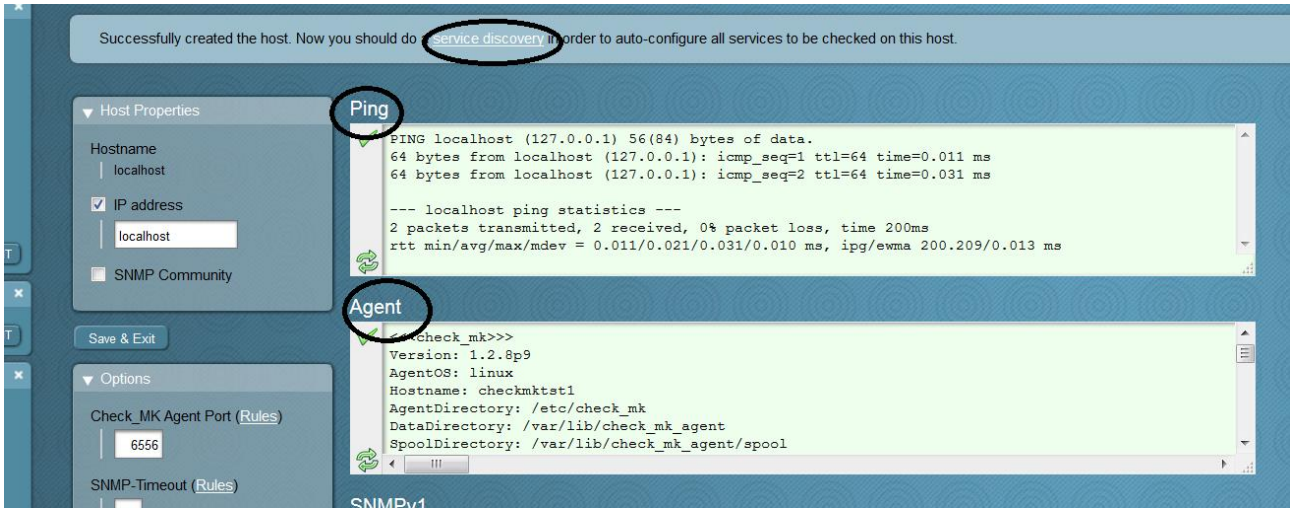
To add, remove a device, use *WATO: Hosts*



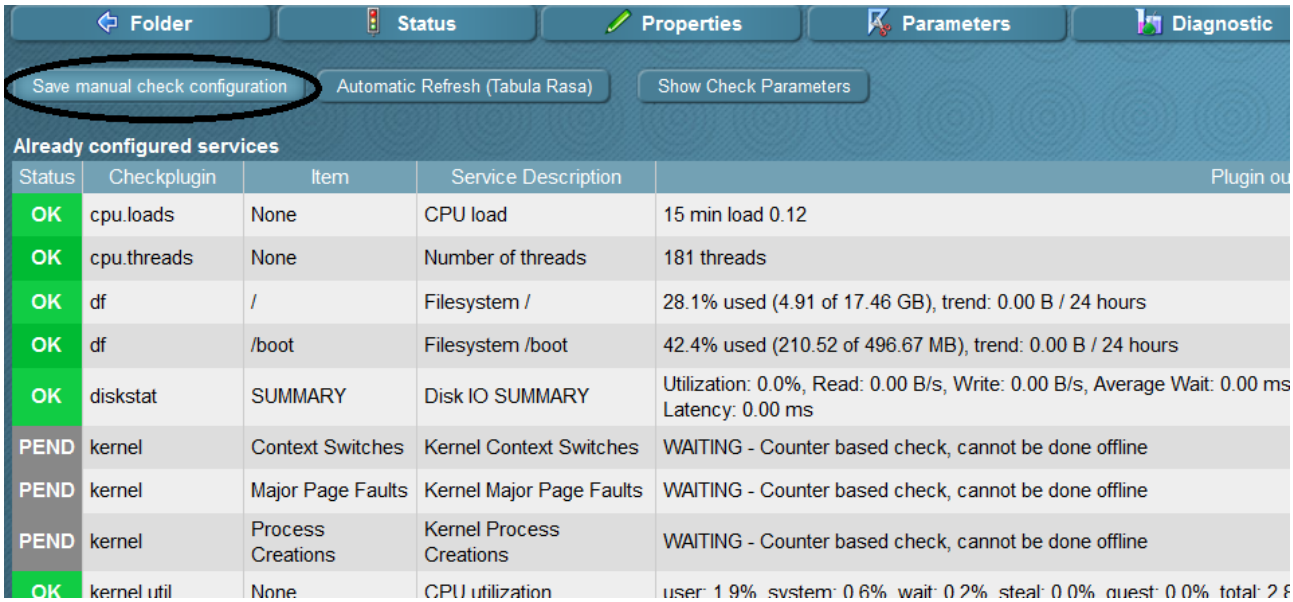
As *Agent Type*, leave the default *Check_MK_Agent (Server)*



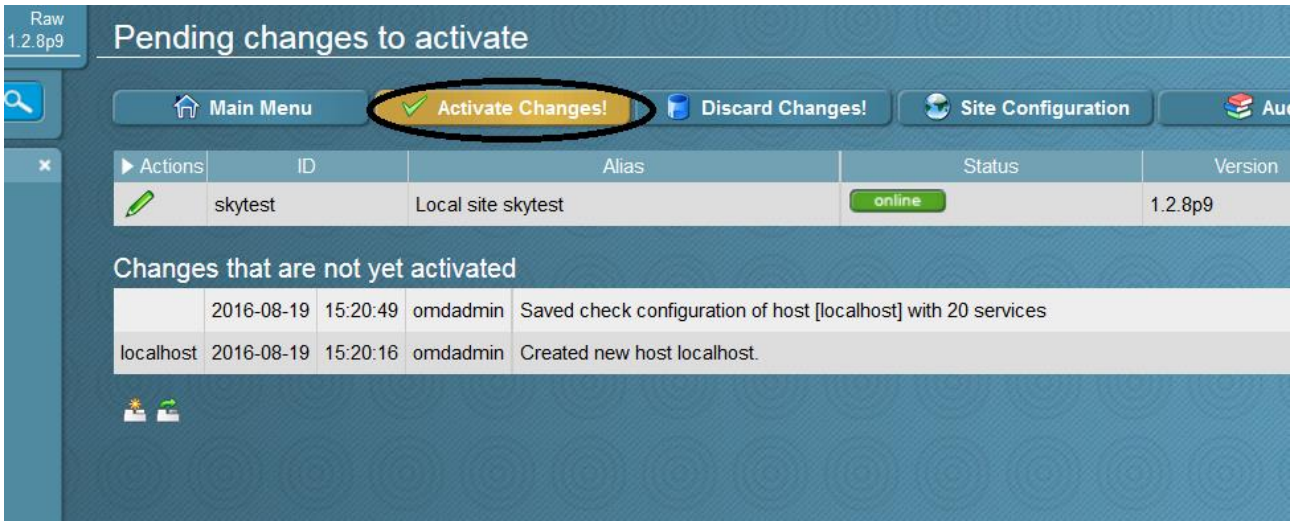
This is the output of an agent that is replying correctly



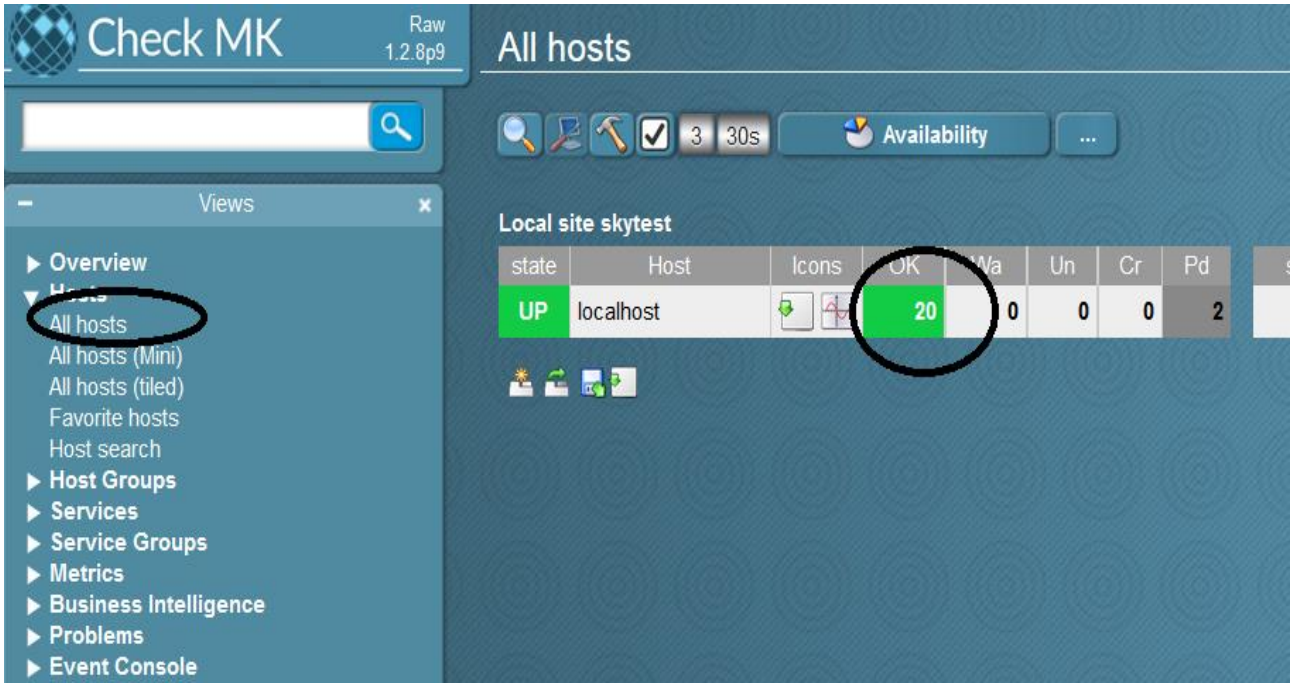
Click on *Service Discovery* and *Save manual check configuration*

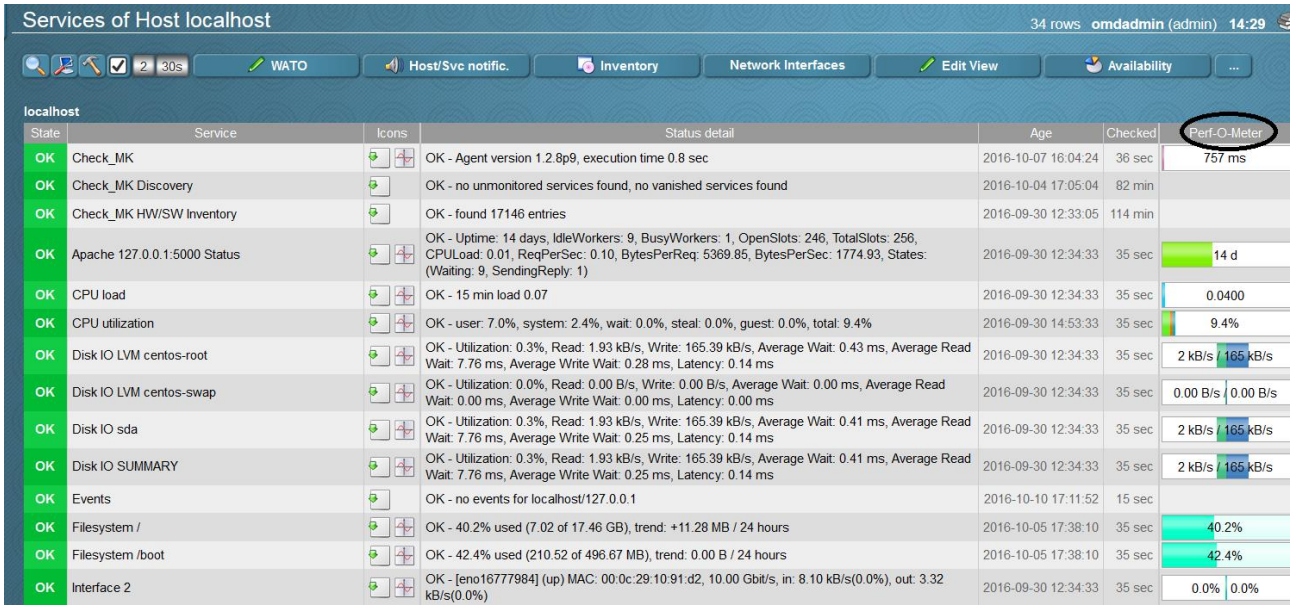


Apply changes

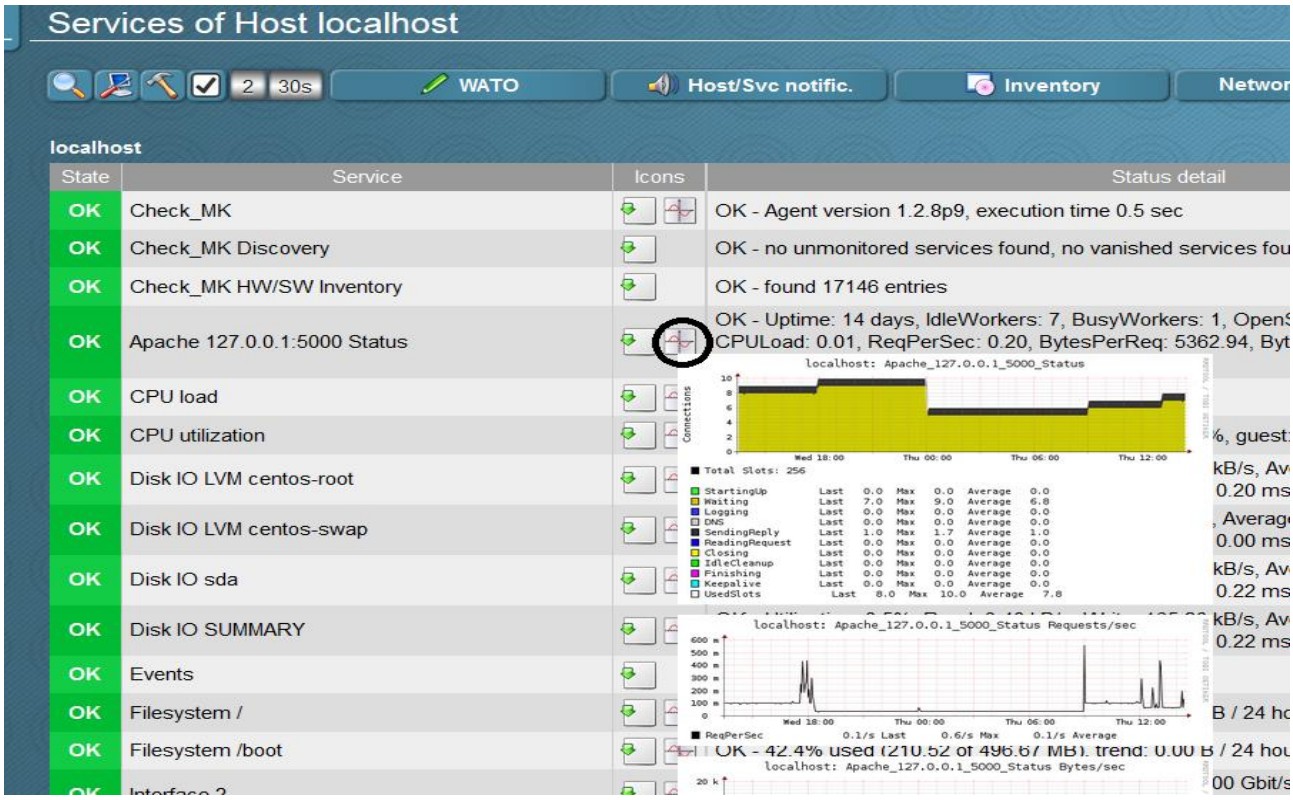


After a couple of minutes, we'll be able to see the list of all services the agent is monitoring on the host, along with their full status and their 'Perf-O-Meters' that show performance metrics where applicable.

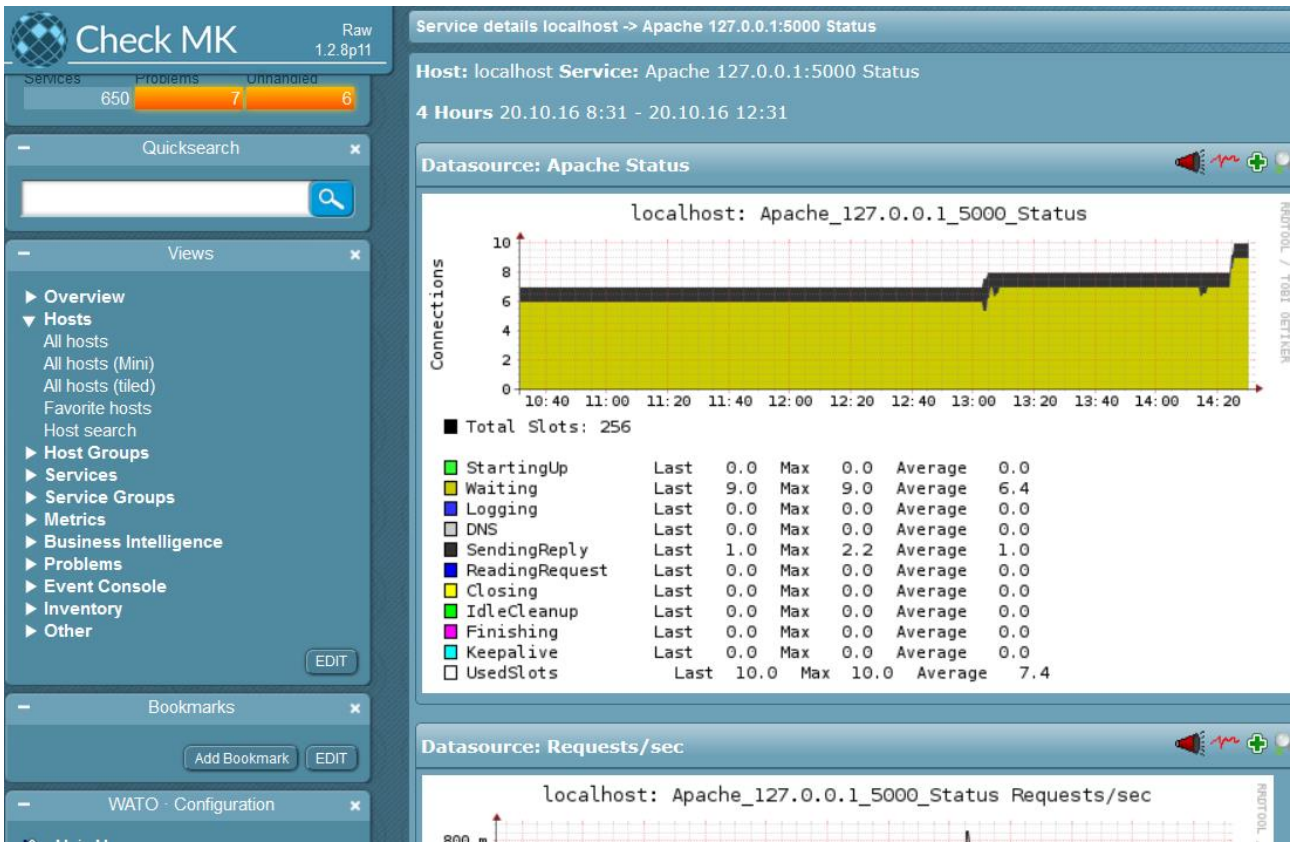




A preview of detailed Performance Graphs (rrd) are accessible hovering the mouse over the graph icon.

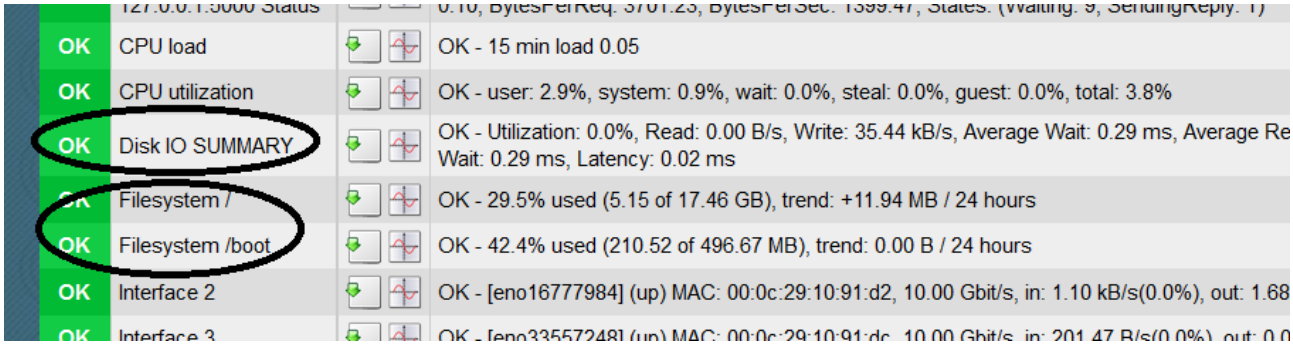


Clicking on the icon causes the graphs to be displayed in a new window

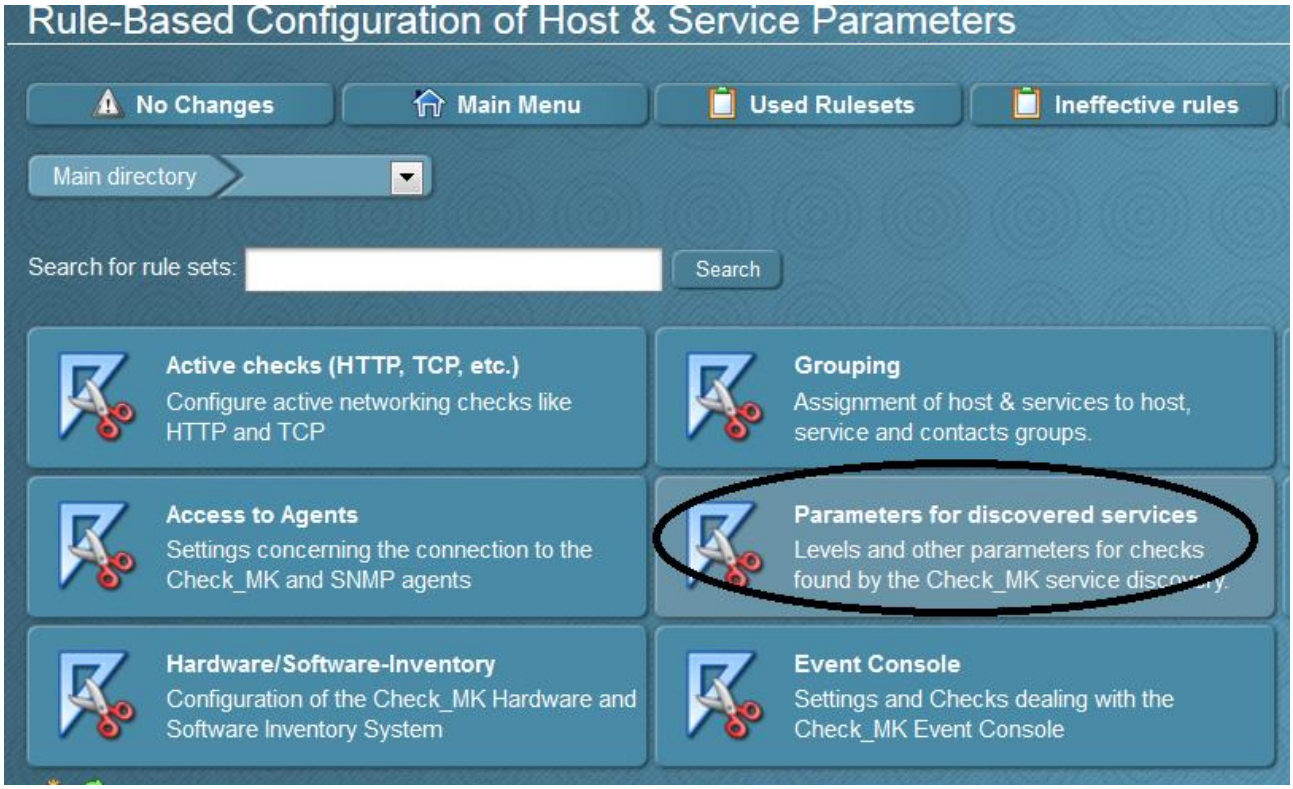


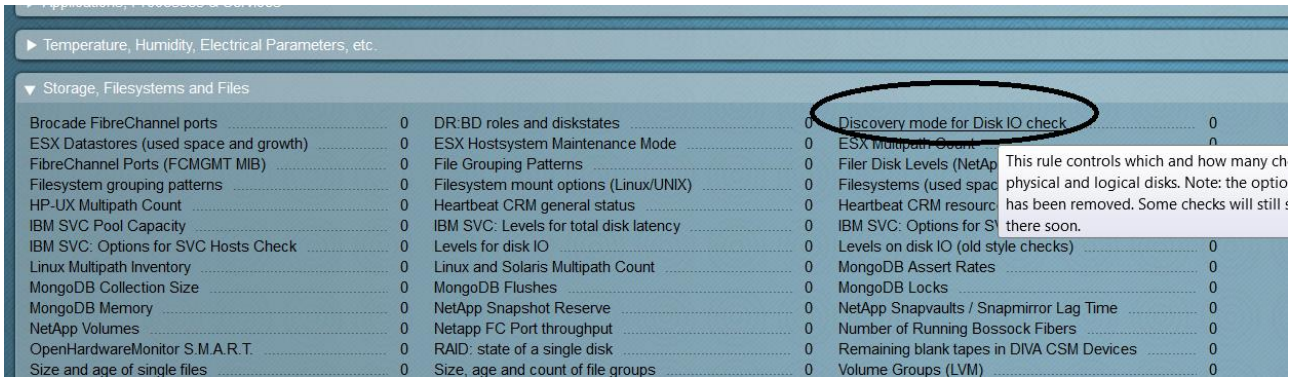
File System Monitoring

By default, Check_MK creates a service for every filesystem and a specific service called *Disk IO Summary* that measures the throughput of block devices (disks) on Linux hosts. You can either have a single check for every single disk or a summary check (which is the default) summing up the throughput of all disks together.

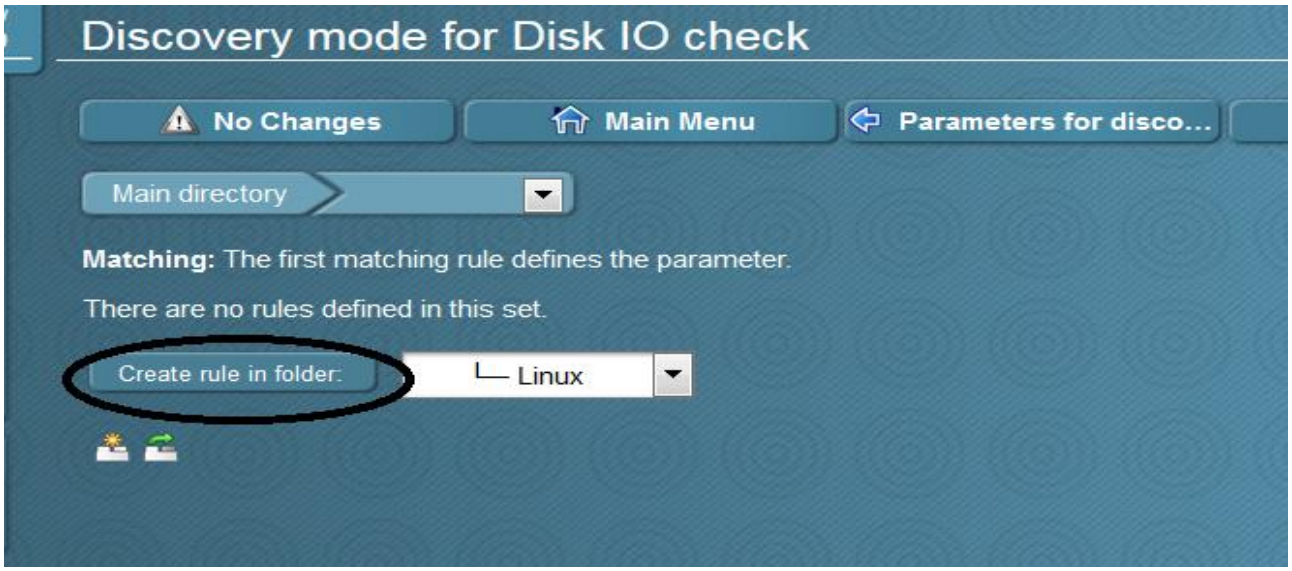


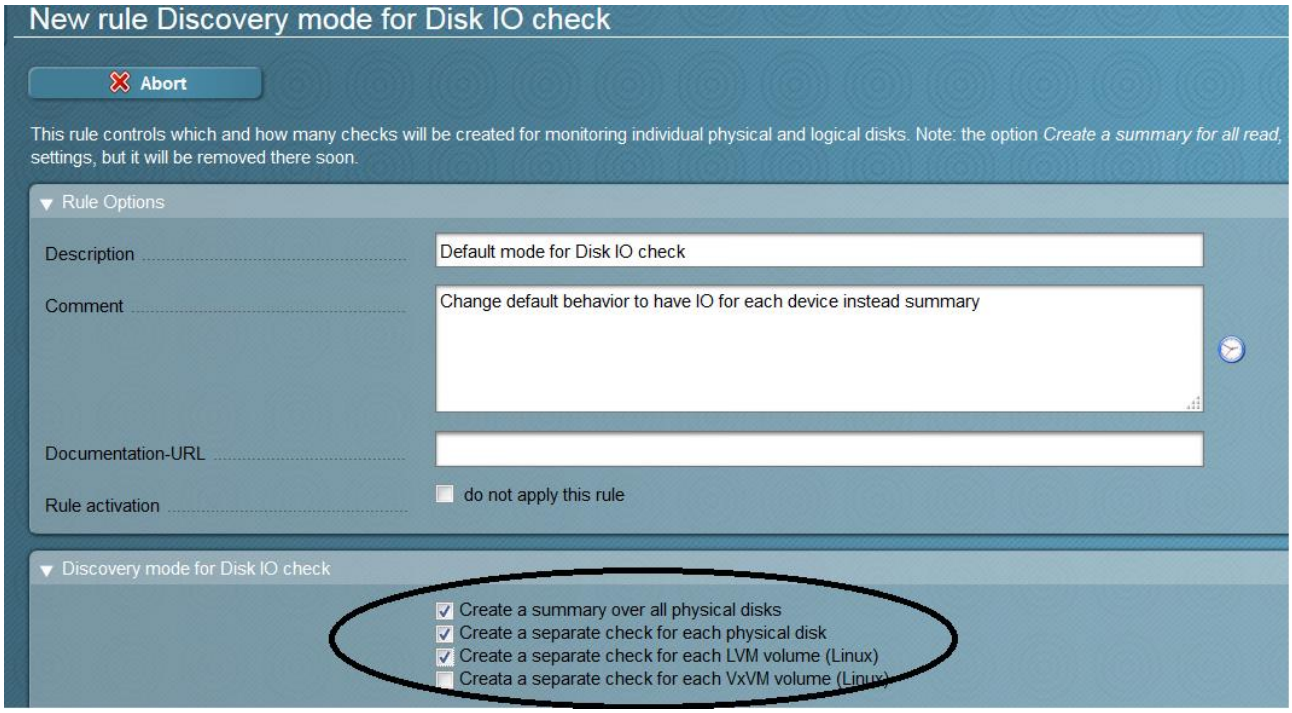
It's easy to change the default behavior as follows. Using *WATO: Host & Service Parameters, Parameters for discovered services, Storage, Filesystems and Files*



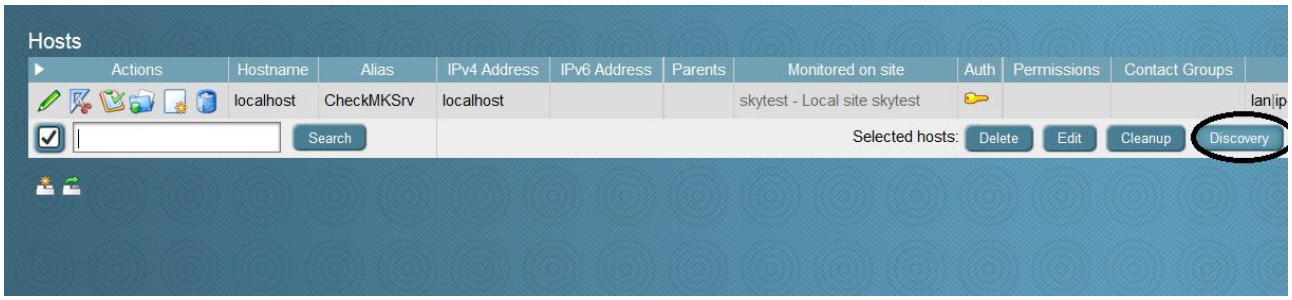


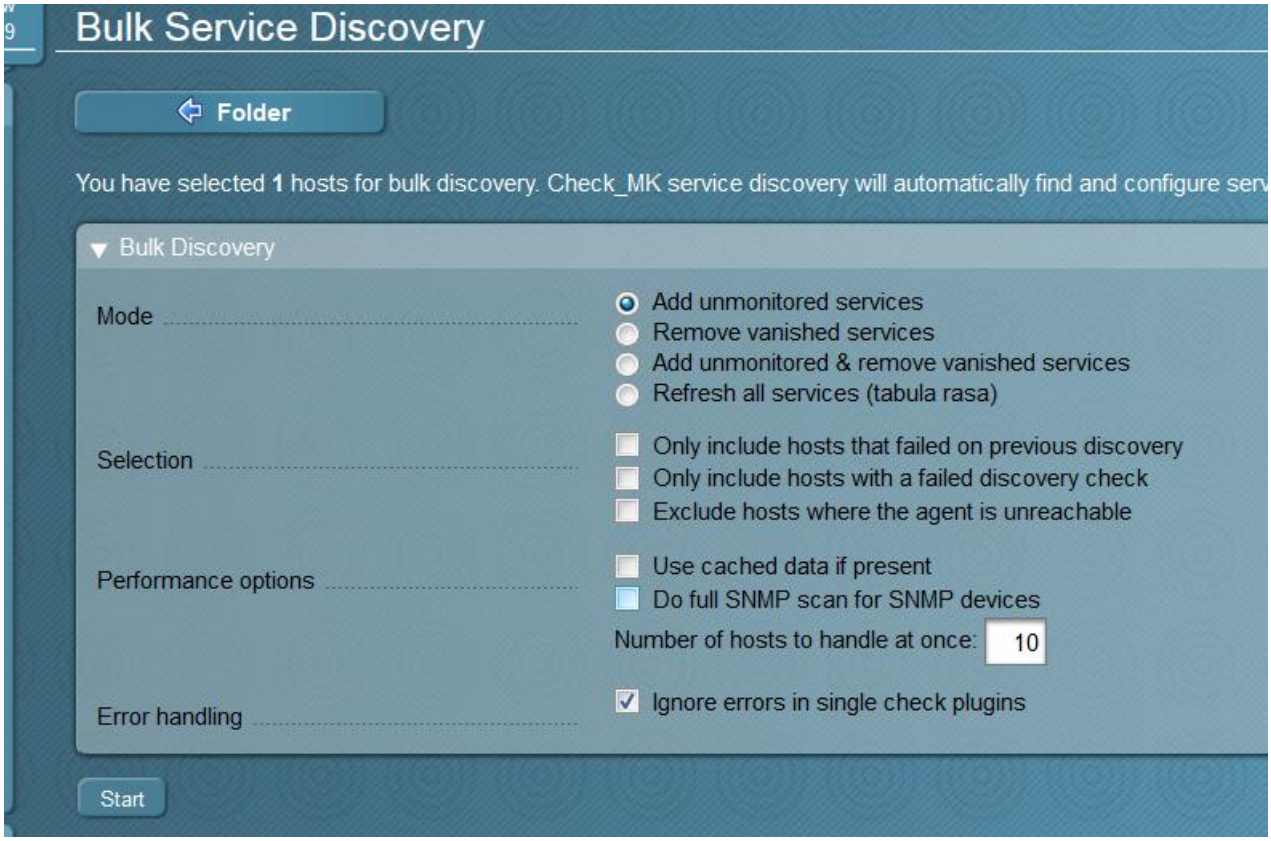
Create a new rule

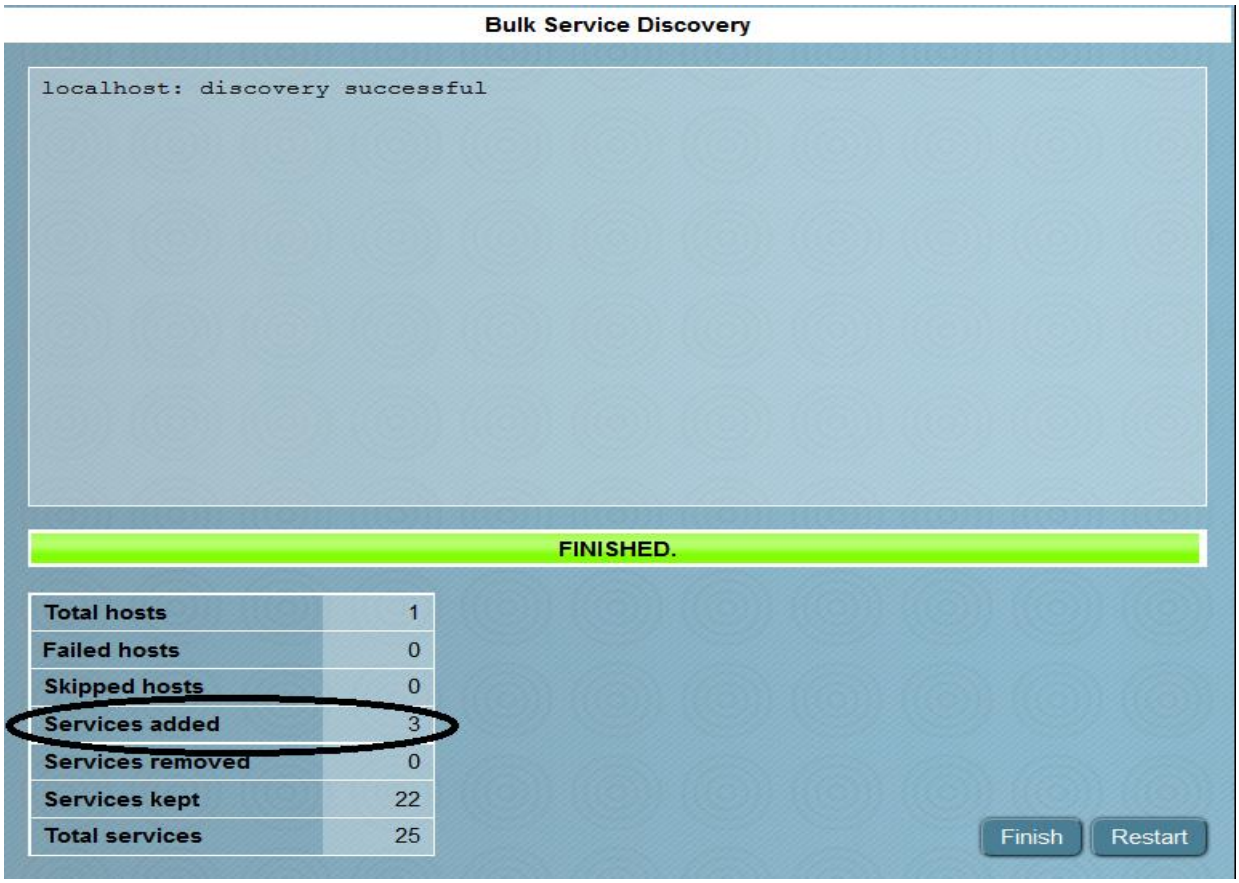




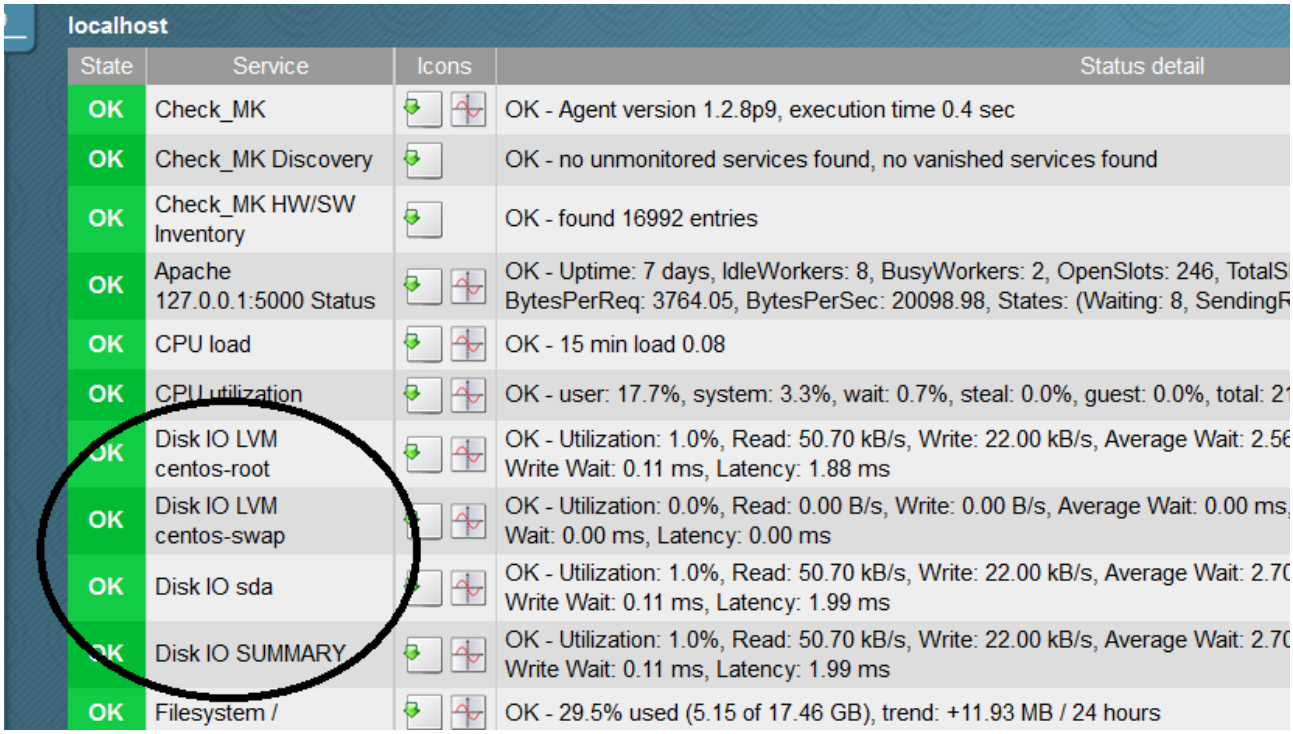
Do a Service discovery to add new services







Click Finish and apply changes. The filesystem output should change to something like this:



Linux Process Monitoring

Monitoring of Linux processes is achieved using the ***ps*** plugin. This looks through the list of current running processes for those matching a certain name or regular expression (and optionally for those owned by a certain user). It's also possible to define thresholds for the number of running processes as well for cpu or memory usage etc.
If you also need performance data, the ***ps.perf*** plugin does exactly the same as **ps** but, as might be expected, outputs performance data.

Let's monitor the **httpd** process:

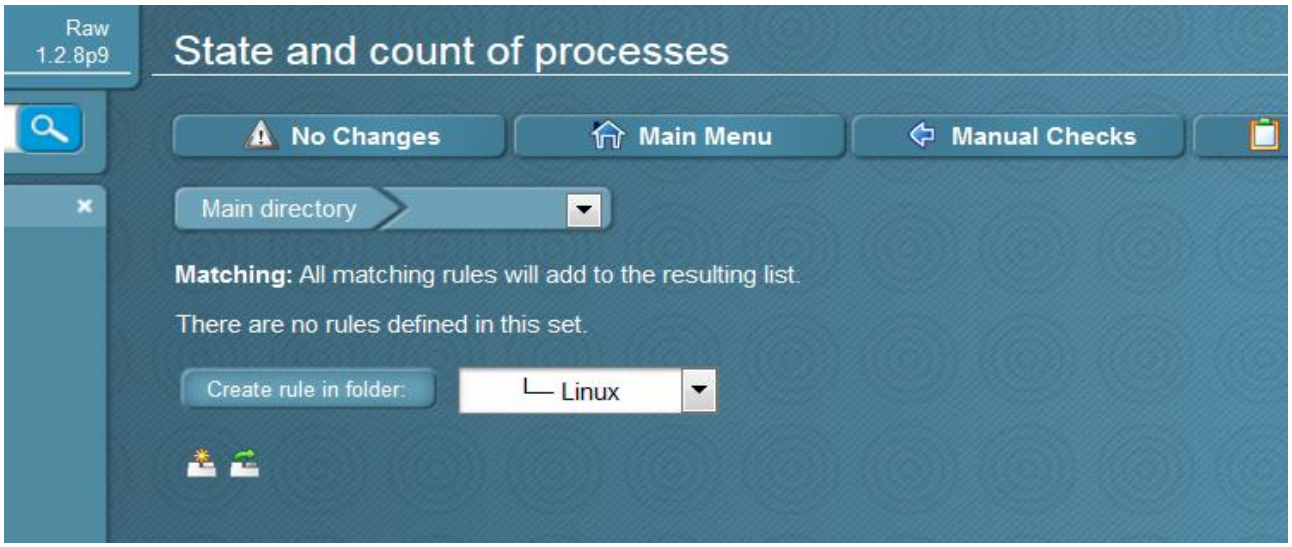
As a first step I suggest checking the specific command line arguments of the process from the shell:

```
[root@checkmktst1 ~]# ps -ef | grep httpd
mysite    395 12169  0 11:10 ?       00:00:01 /usr/sbin/httpd -f /omd/sites/mysite/etc/apache/apache.conf
apache    396  1029  0 11:10 ?       00:00:00 /usr/sbin/httpd -DFOREGROUND
root     1029     1  0 Aug26 ?       00:00:13 /usr/sbin/httpd -DFOREGROUND
apache   1928  1029  0 Aug29 ?       00:00:02 /usr/sbin/httpd -DFOREGROUND
apache   3606  1029  0 11:23 ?       00:00:00 /usr/sbin/httpd -DFOREGROUND
apache   4427  1029  0 11:25 ?       00:00:00 /usr/sbin/httpd -DFOREGROUND
apache   7587  1029  0 11:34 ?       00:00:00 /usr/sbin/httpd -DFOREGROUND
apache   8519  1029  0 11:36 ?       00:00:00 /usr/sbin/httpd -DFOREGROUND
apache   8944  1029  0 11:37 ?       00:00:00 /usr/sbin/httpd -DFOREGROUND
apache   9086  1029  0 11:37 ?       00:00:00 /usr/sbin/httpd -DFOREGROUND
mysite  12169     1  0 Aug26 ?       00:00:10 /usr/sbin/httpd -f /omd/sites/mysite/etc/apache/apache.conf
root    14266  3560  0 11:55 pts/0    00:00:00 grep --color=auto httpd
apache  19129  1029  0 Aug31 ?       00:00:00 /usr/sbin/httpd -DFOREGROUND
apache  19486  1029  0 10:24 ?       00:00:00 /usr/sbin/httpd -DFOREGROUND
mysite  19513 12169  0 10:24 ?       00:00:03 /usr/sbin/httpd -f /omd/sites/mysite/etc/apache/apache.conf
apache  19514  1029  0 10:24 ?       00:00:00 /usr/sbin/httpd -DFOREGROUND
mysite  19545 12169  0 10:24 ?       00:00:02 /usr/sbin/httpd -f /omd/sites/mysite/etc/apache/apache.conf
mysite  19579 12169  0 10:24 ?       00:00:02 /usr/sbin/httpd -f /omd/sites/mysite/etc/apache/apache.conf
mysite  19686 12169  0 00:00 ?       00:00:00 /usr/sbin/httpd -f /omd/sites/mysite/etc/apache/apache.conf
mysite  19688 12169  0 00:00 ?       00:00:04 /usr/sbin/httpd -f /omd/sites/mysite/etc/apache/apache.conf
mysite  19689 12169  0 00:00 ?       00:00:02 /usr/sbin/httpd -f /omd/sites/mysite/etc/apache/apache.conf
mysite  19690 12169  0 00:00 ?       00:00:05 /usr/sbin/httpd -f /omd/sites/mysite/etc/apache/apache.conf
mysite  19978 12169  0 00:01 ?       00:00:04 /usr/sbin/httpd -f /omd/sites/mysite/etc/apache/apache.conf
mysite  27189 12169  0 10:51 ?       00:00:02 /usr/sbin/httpd -f /omd/sites/mysite/etc/apache/apache.conf
mysite  31447 12169  0 11:04 ?       00:00:02 /usr/sbin/httpd -f /omd/sites/mysite/etc/apache/apache.conf
```

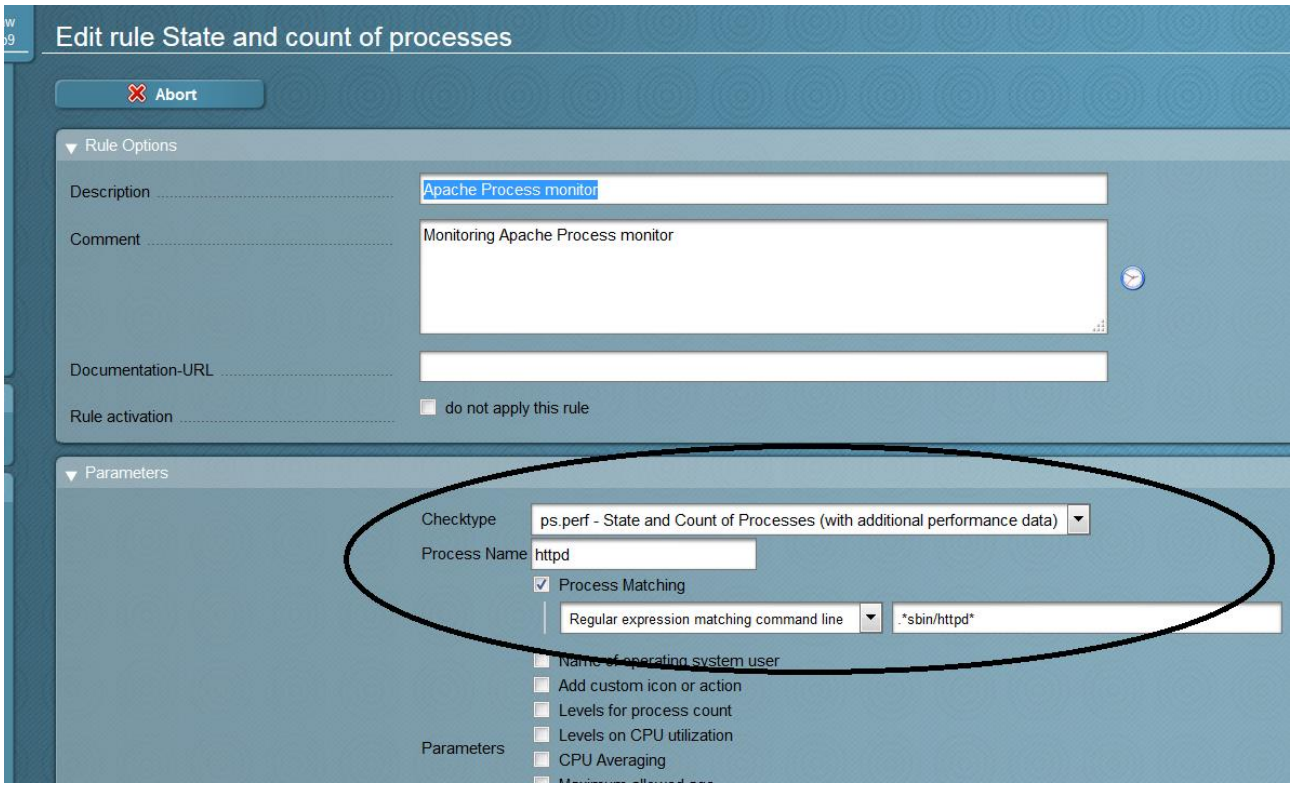
Count how many *httpd* processes are running:

```
[root@checkmktst1 ~]# pidof httpd | wc -w
24
```

Now we can define the service check for the Apache process using the GUI: WATO: *Manual Checks, Application Processes & Services, State and count processes*



Because we need to get all processes that have the string *"/sbin/httpd *"*, we need to use a simple regular expression. The plugin homepage guides us through these steps.



Don’t forget to save and apply changes!

Following this we should see that the new service check has been applied and useful performance graphs are being generated.

OK	Number of threads		OK - 184 threads
OK	OMD [redacted]apache		OK - 0.07 Requests/s, 0.00 Seconds serving/s, 143.46 B Sent/s
OK	OMD [redacted]performance		OK - Host Checks: 0.2/s, Service Checks: 11.9/s, Process Creations: 0.5/s, Livestatus Connects: 0.1/s, Livestatus Requests: 0.2/s, Log Messages: 0.0/s, 11 Hosts, 703 Services, Core version: 3.5.0, Livestatus version: 1.2.8p13
OK	OMD [redacted]status		OK - running
WARN	Postfix Queue		WARN - deferred queue length is 17 (Levels at 10/20) WARN, active queue length is 0
OK	Process httpd		OK - 24 processes 6766.6 MB virtual, 631.2 MB resident, 0.0% CPU, youngest running for 87 min oldest running for 18 hours
OK	[redacted]CUSTOMSCRIPT_TEST_Filecount_/tmp		OK - 8 files in /tmp
CRIT	[redacted]CUSTOMSCRIPT_TEST_Filecount_/var/log		CRIT - CRITICAL - 60 files in /var/log
OK	[redacted]_TST_TCPCHECK_SMTP		TCP OK - 0.001 second response time on 127.0.0.1 port 25
OK	TCP Connections		OK - ESTABLISHED: 2, TIME_WAIT: 5, LISTEN: 10
OK	Uptime		OK - up since Wed Oct 26 16:38:30 2016 (0d 18:40:01)

Log Files

Logfiles on Linux are monitored using the *logwatch* extension for the check_mk_agent.

-Copy *mk_logwatch* in the plugin directory.

cp /opt/omd/versions/1.2.8p9.cre/share/check_mk/agents/plugins/mk_logwatch /usr/lib/check_mk_agent/plugins

Create the file */etc/check_mk/logwatch.cfg* with the following text:

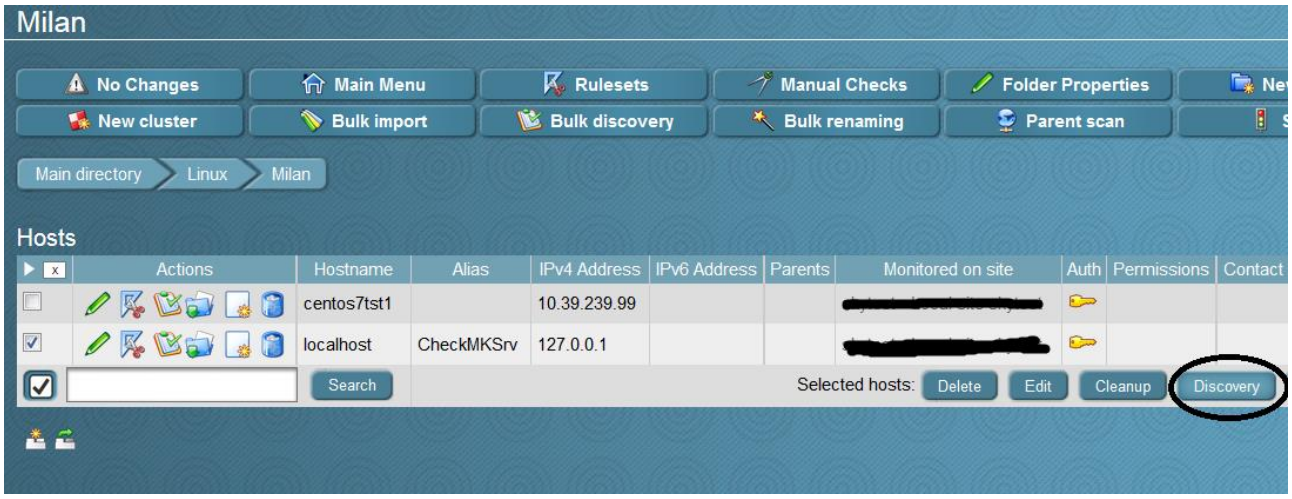
```
/var/log/messages
 C Error*
 R TEST: This is a fake error, monitoring a logfile just as test \1
```

The first line specified the text file we want to monitor; the second means that if the agent finds the expression “Error” (followed by any words) a critical error will be created. The last creates a rewrite rule, customizing the message that will be displayed within the GUI

-Restart the agent

service xinetd restart

-Do a discovery on localhost so that the new check will be automatically added



-Activate changes

-Do a test

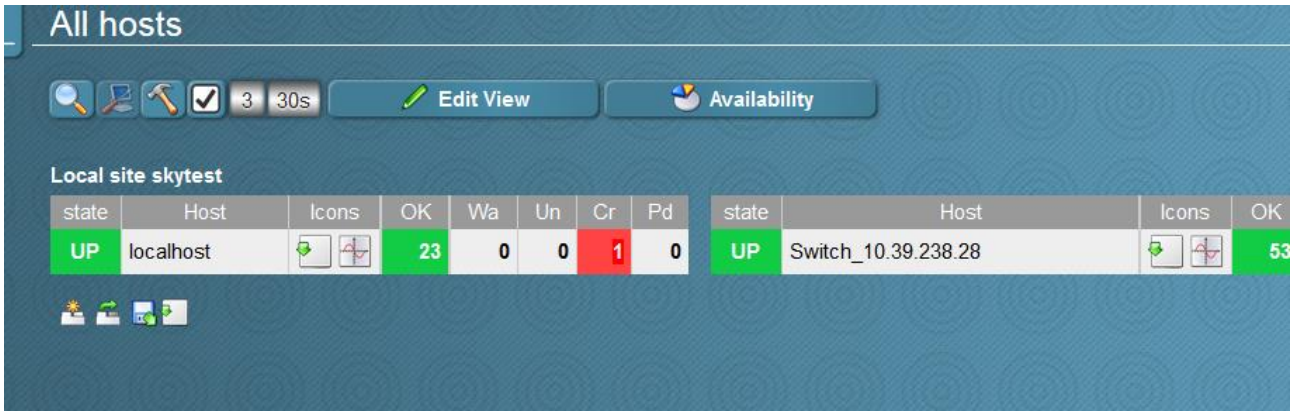
echo "Error" >> /var/log/messages

-Test the agent from command line

su - mysite

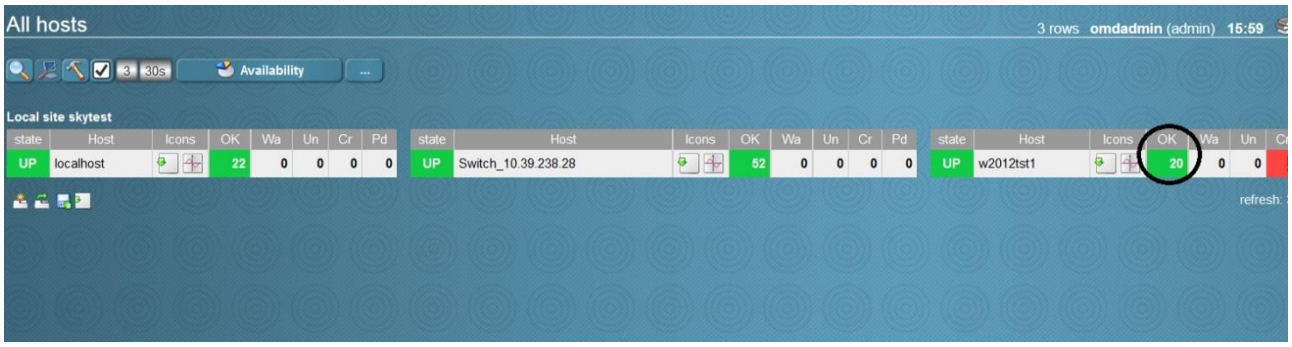
<<<job>>>
<<<local>>>
<<<logwatch>>>
[[[/var/log/messages]]]
Aug 26 15:35:01 checkmktst1 systemd: Created slice user-986.slice.
Aug 26 15:35:01 checkmktst1 systemd: Starting user-986.slice.
Aug 26 15:35:01 checkmktst1 systemd: Started Session 409 of user mysite.
Aug 26 15:35:01 checkmktst1 systemd: Starting Session 409 of user mysite.
Aug 26 15:35:01 checkmktst1 systemd: Started Session 410 of user mysite.
Aug 26 15:35:01 checkmktst1 systemd: Starting Session 410 of user mysite.
Aug 26 15:35:01 checkmktst1 systemd: Started Session 411 of user mysite.
Aug 26 15:35:01 checkmktst1 systemd: Starting Session 411 of user mysite.
Aug 26 15:35:02 checkmktst1 systemd: Removed slice user-986.slice.
Aug 26 15:35:02 checkmktst1 systemd: Stopping user-986.slice.
C TEST: This is a fake error, monitoring a logfile just as test \1
Aug 26 15:35:09 checkmktst1 su: (to mysite) root on pts/0

-Loot at WATO to check if the CRITICAL has been generated



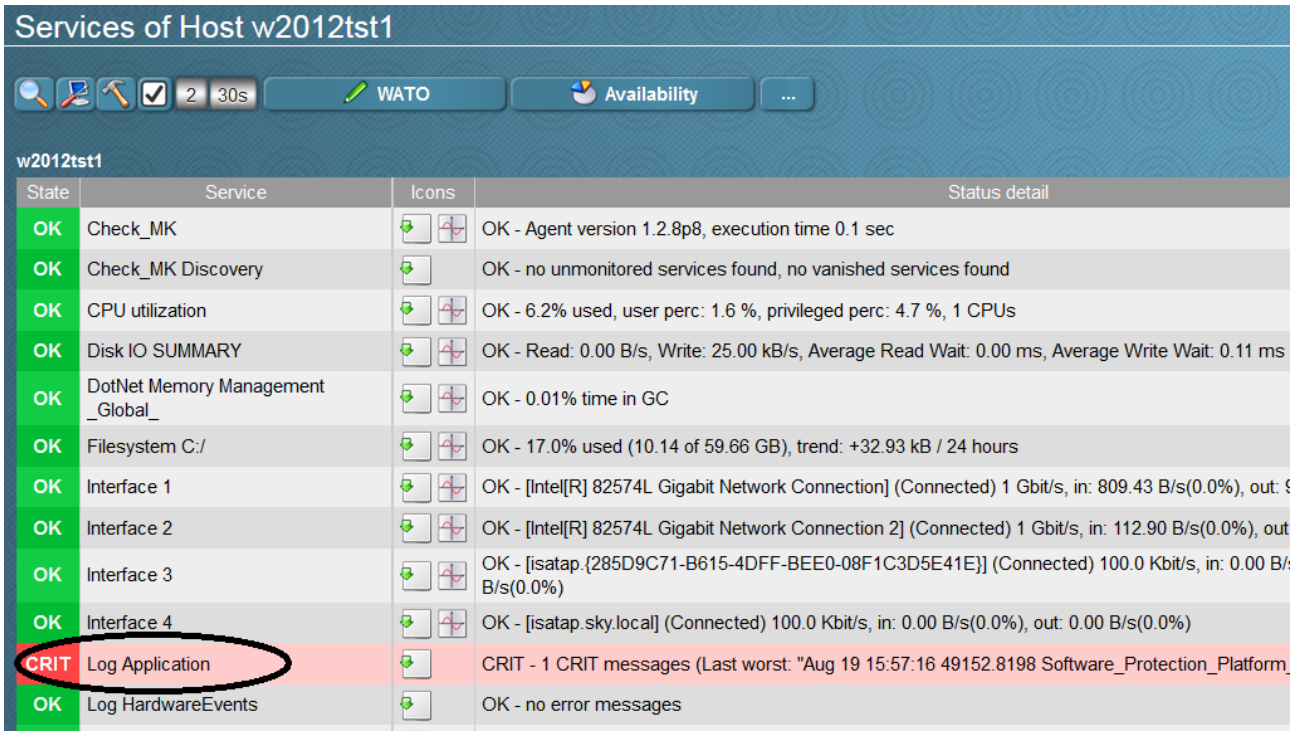
Windows Devices

Download and Install the *check_mk_agent.msi* on the Windows server. The same steps that we carried out for Linux also apply to Windows devices.



Windows Event Viewer

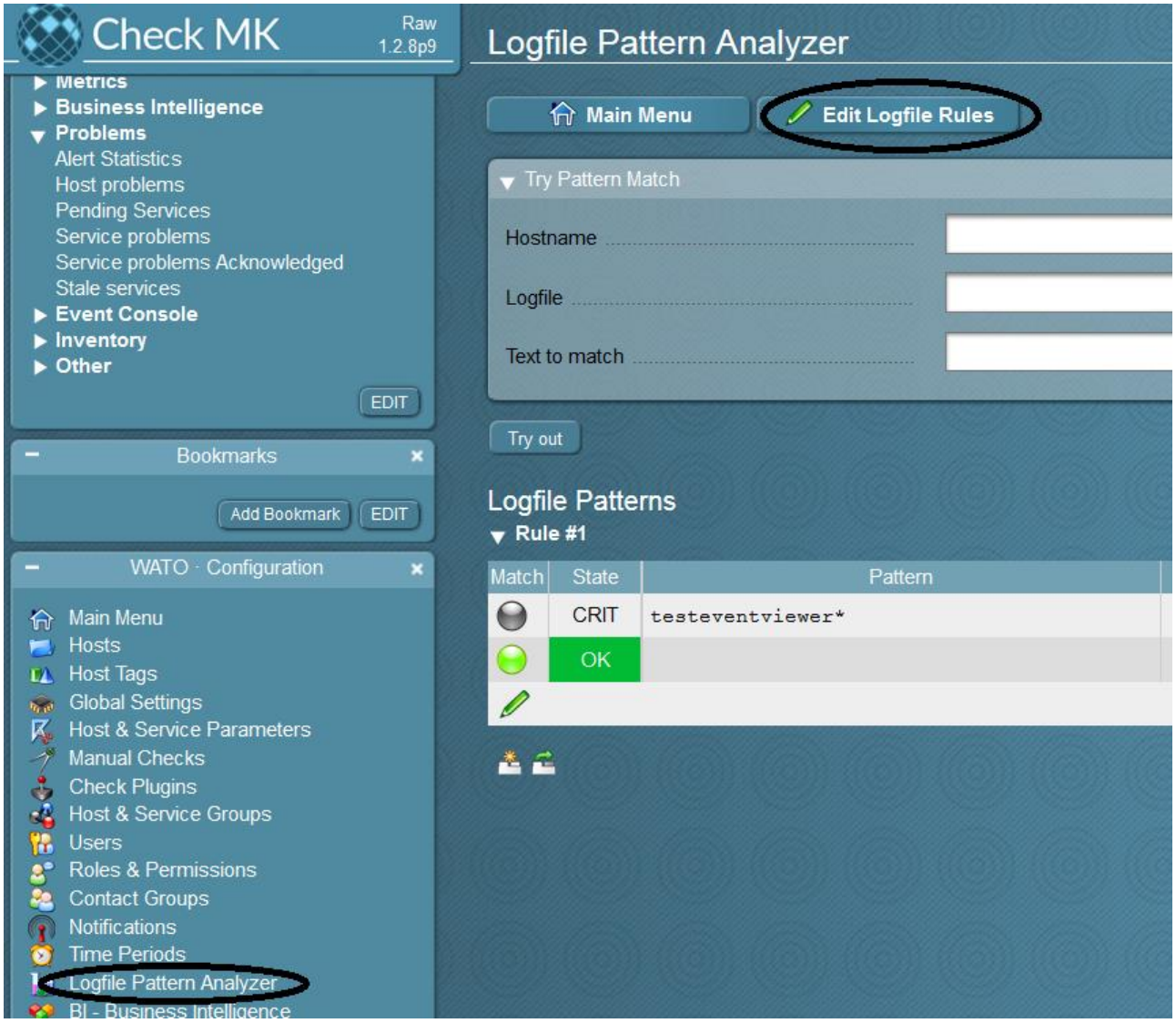
By default the Windows agent sends all non-informational messages to the Check_MK server. We can see here that Check_MK automatically detected an error in the Windows Event Log.



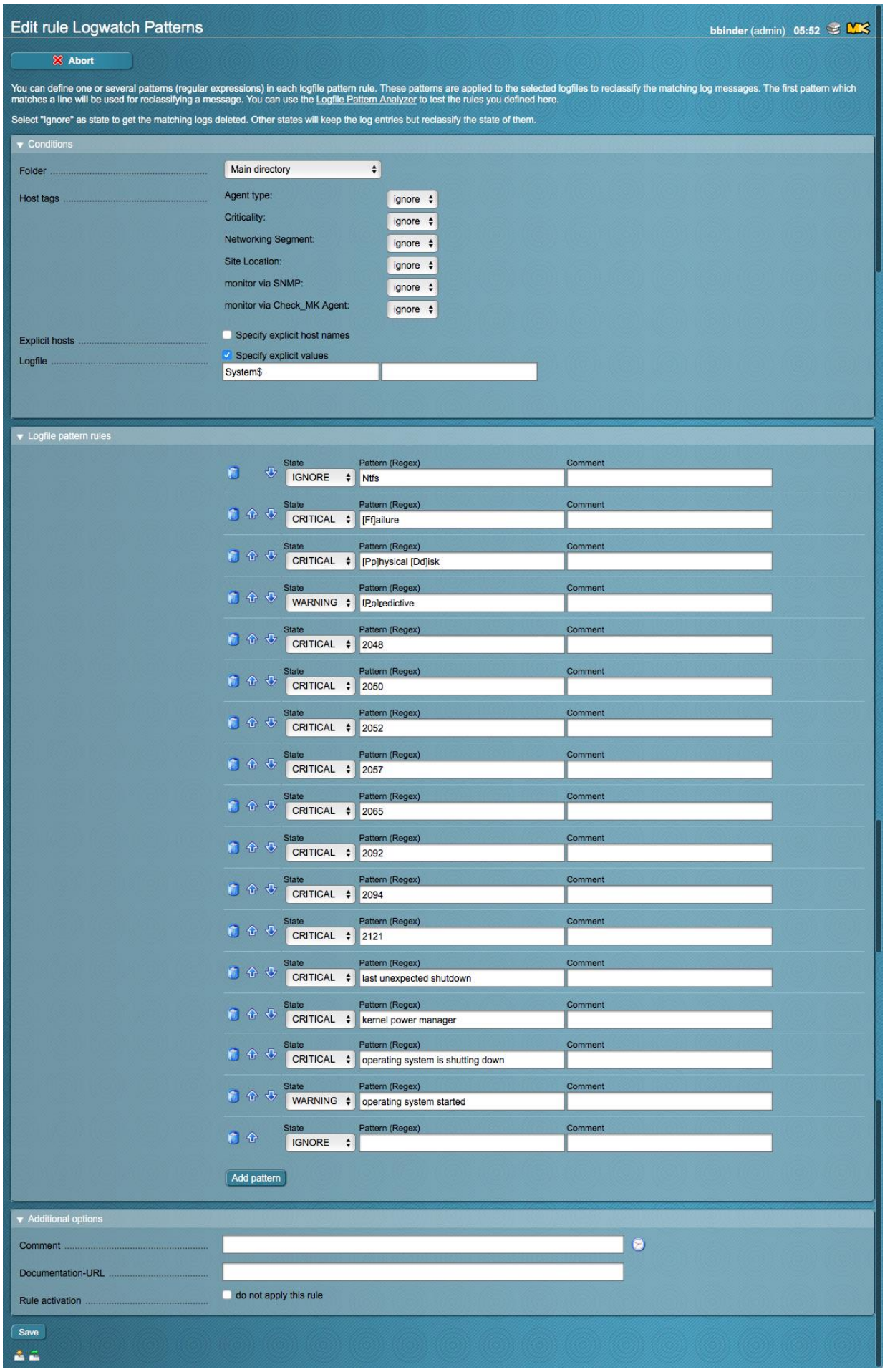
Since the agent is completely configuration-less, it doesn't do specific filtering of events. It simply looks for messages of type **Warning** or **Error**. This behavior can be changed by creating a file called *check_mk.ini* in the agent directory but, in my opinion, this isn't the best way - if you have hundreds of servers, re-deploying the configuration file and restarting all agents can be a pain. A better approach is to create "centralized" rules which specify a list of "windows event id" or strings for each "Windows event log" that you consider critical. I know that this solution requires some time to optimize, but with a bit of experience

(and Google searching!), it can have excellent results. For example, in my environment I added some rules relating to Oracle (e.g. “ORA-RAC”), MSSQL (e.g. cluster failed) etc.

Click on *Logfile Pattern Analyzer*, *Edit Logfile Rules*

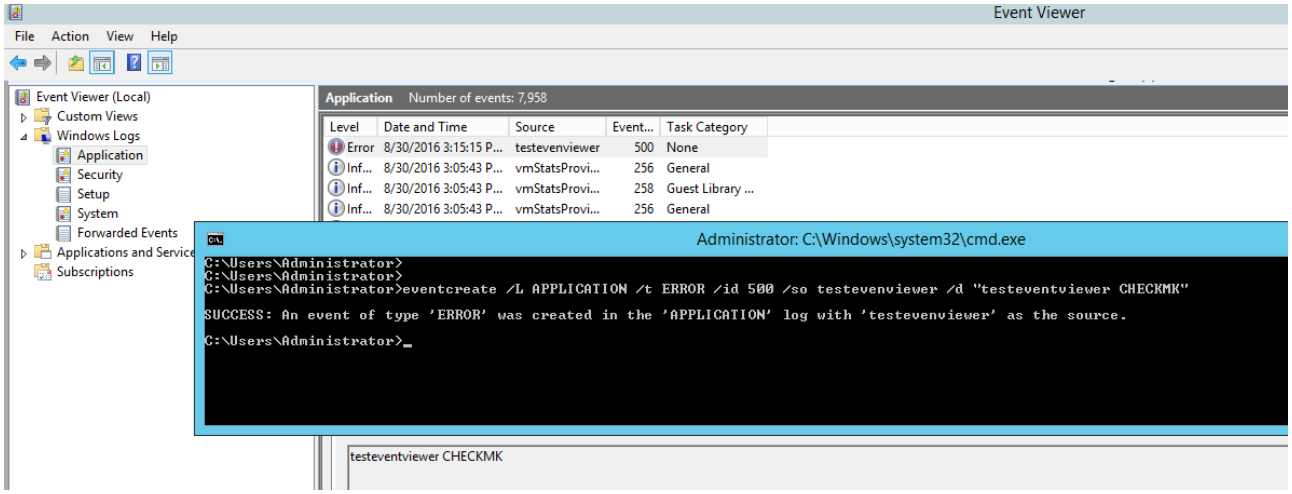


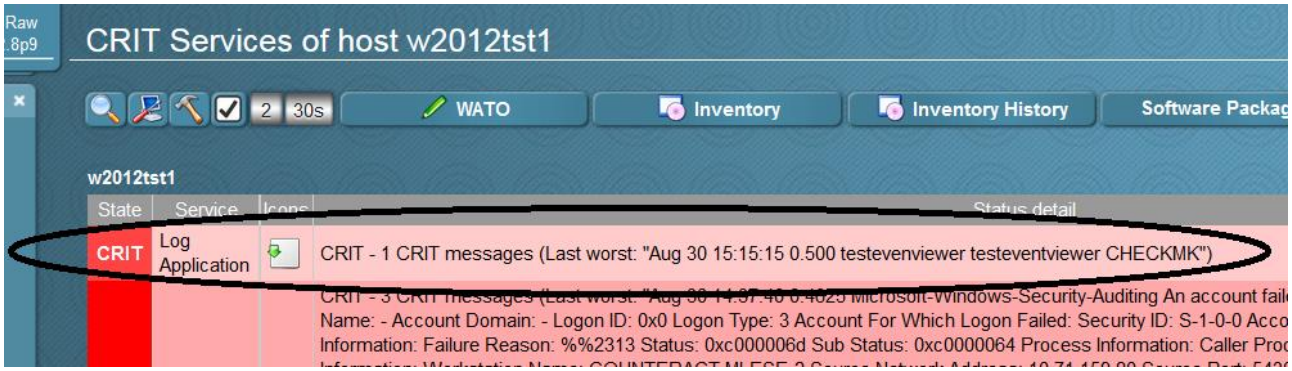
In this picture you can see the rule for *System event log*. **Please pay attention to the order of the rules!** See that ignore is on the bottom, and then I’m adding values on the top as they fire from the top down. Note the *WARNING* or *CRITICAL* entries I’m making for the specific entries I’ve added.


Edit rule Logwatch Patterns
bbinder (admin) 05:52
Abort
You can define one or several patterns (regular expressions) in each logfile pattern rule. These patterns are applied to the selected logfiles to reclassify the matching log messages. The first pattern which matches a line will be used for reclassifying a message. You can use the Logfile Pattern Analyzer to test the rules you defined here.
Select "Ignore" as state to get the matching logs deleted. Other states will keep the log entries but reclassify the state of them.
Conditions
Folder
Main directory
Host tags
Agent type: ignore
Criticality: ignore
Networking Segment: ignore
Site Location: ignore
monitor via SNMP: ignore
monitor via Check_MK Agent: ignore
Explicit hosts
Specify explicit host names
Logfile
Specify explicit values
System$
Logfile pattern rules
State Pattern (Regex) Comment
IGNORE Ntfs
CRITICAL [Ff]ailure
CRITICAL [Pp]hysical [Dd]isk
WARNING [Pp]redictive
CRITICAL 2048
CRITICAL 2050
CRITICAL 2052
CRITICAL 2057
CRITICAL 2065
CRITICAL 2092
CRITICAL 2094
CRITICAL 2121
CRITICAL last unexpected shutdown
CRITICAL kernel power manager
CRITICAL operating system is shutting down
WARNING operating system started
IGNORE
Add pattern
Additional options
Comment
Documentation-URL
Rule activation
do not apply this rule
Save

I test these on a Windows server using *eventcreate* (in this example I'm using the string "testeventviewer CHECKMK" that isn't present in the previous screenshot but for which I've added a rule in my configuration)

```
C:\Users\Administrator>eventcreate /L APPLICATION /t ERROR /id 500 /so testevenviewer /d
"testeventviewer CHECKMK"
```





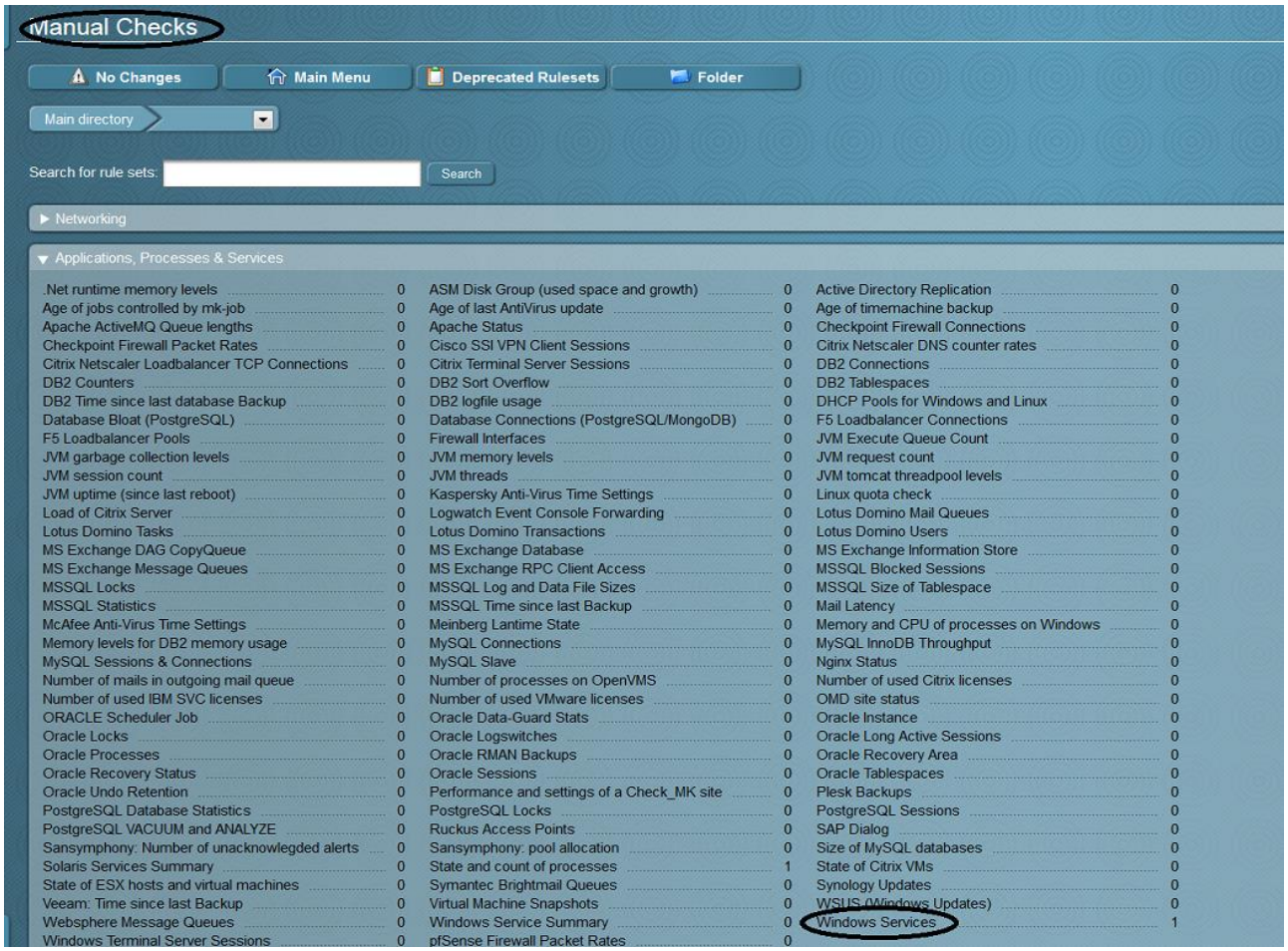
Windows Services

It's also possible to monitor Windows Services but, in this case, we need to specify the name of the services that we would like to monitor. We can specify a list of services that should be monitored on all hosts or just on some of them. In this example I'll show how to monitor "Terminal Server Service" on host *w2012tst1*.
In order to monitor services you first need to determine which services are of interest to you. The easiest way is to look at the raw output of the agent and look for the section <<<services>>>. You can use cmk -d for this:

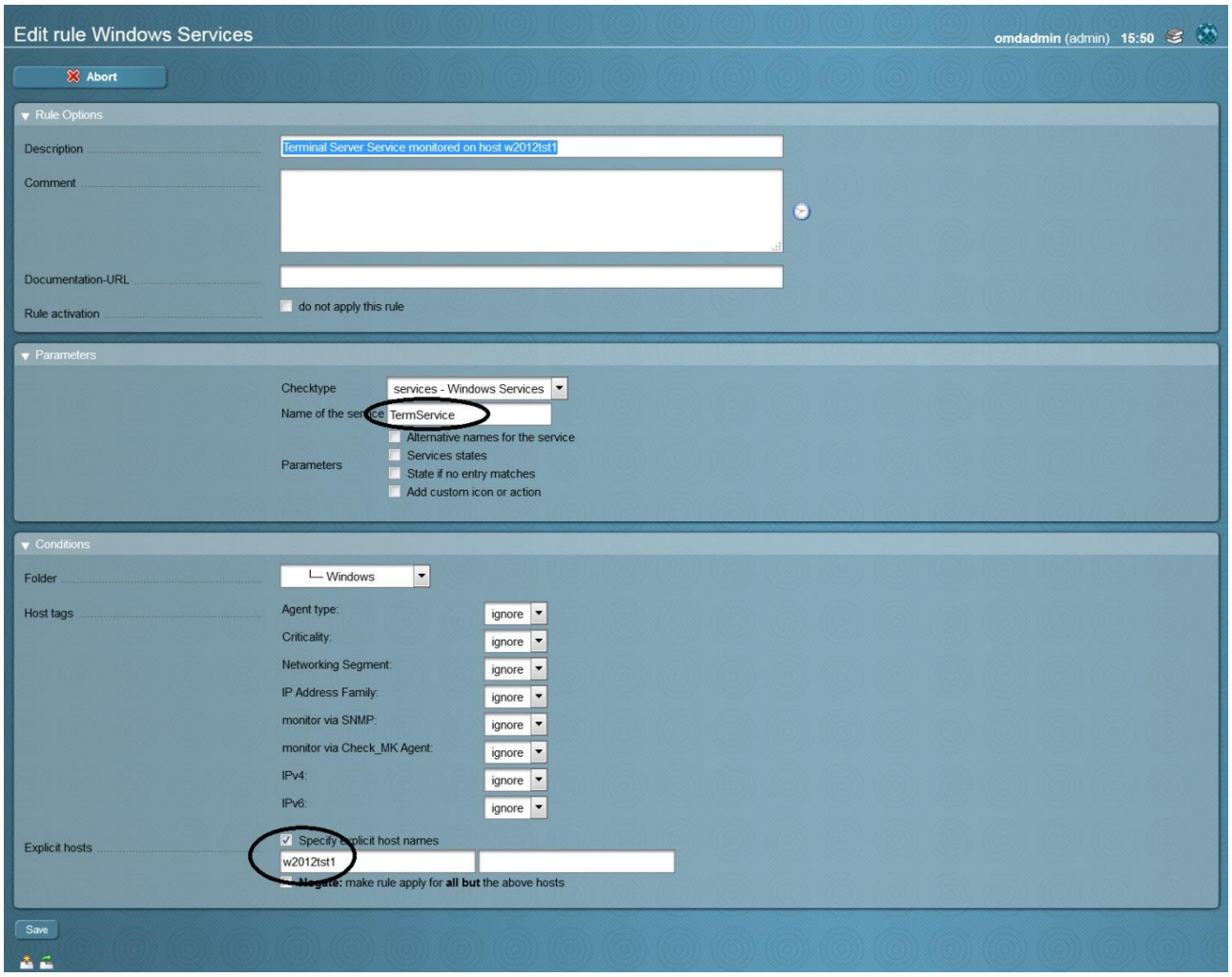
```
OMD[mysite]:~$ cmk -d w2012tst1 | fgrep -A 100 '<<<services>>>' | grep -i running
BFE running/auto Base Filtering Engine
BrokerInfrastructure running/auto Background Tasks Infrastructure Service
CertPropSvc running/demand Certificate Propagation
Check_MK_Agent running/auto Check_MK_Agent
COMSysApp running/demand COM+ System Application
CryptSvc running/auto Cryptographic Services
...
...
...
TermService running/demand Remote Desktop Services
```

The first column of the output is the exact internal name of the service. Let's say you want to check if TermService (Windows Terminal Server) is running on host w2012tst1.

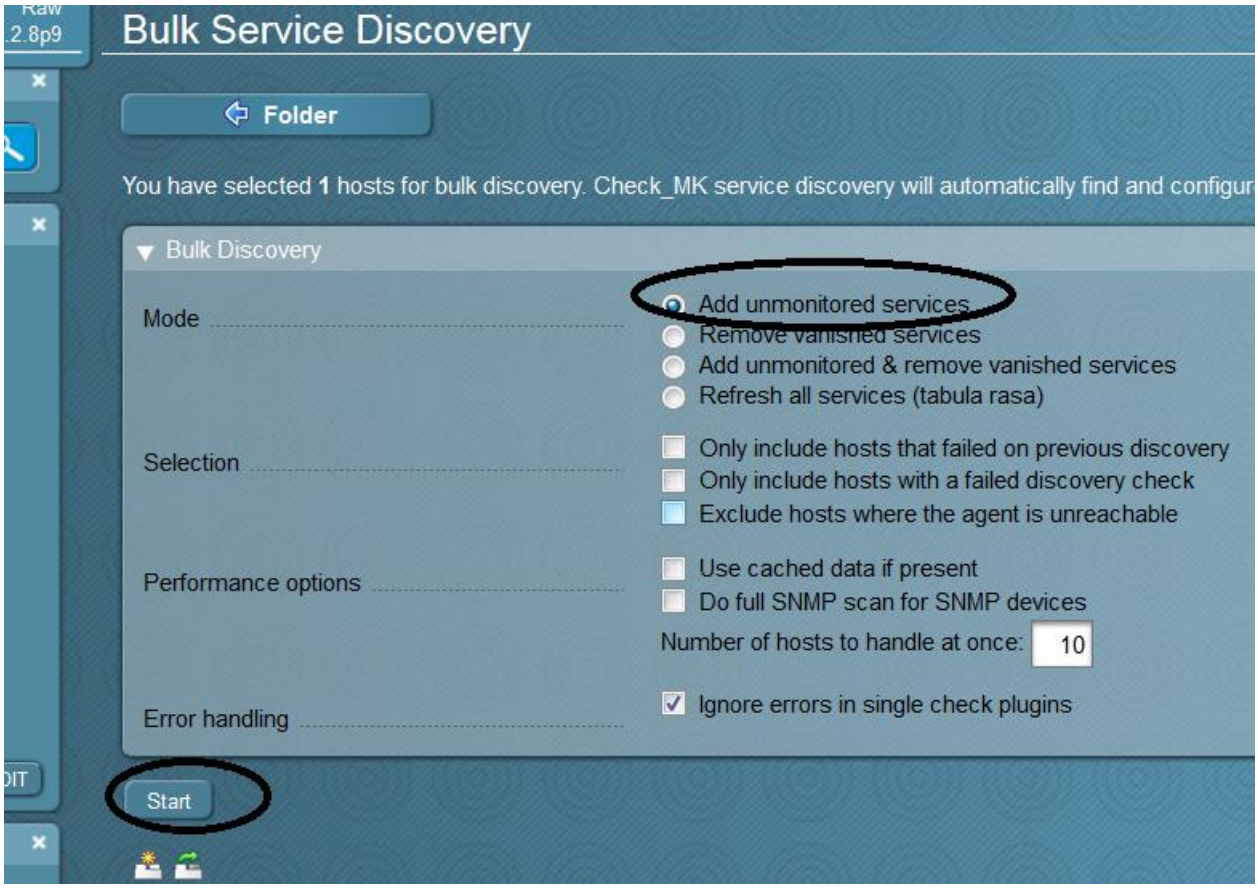
WATO, Manual Checks

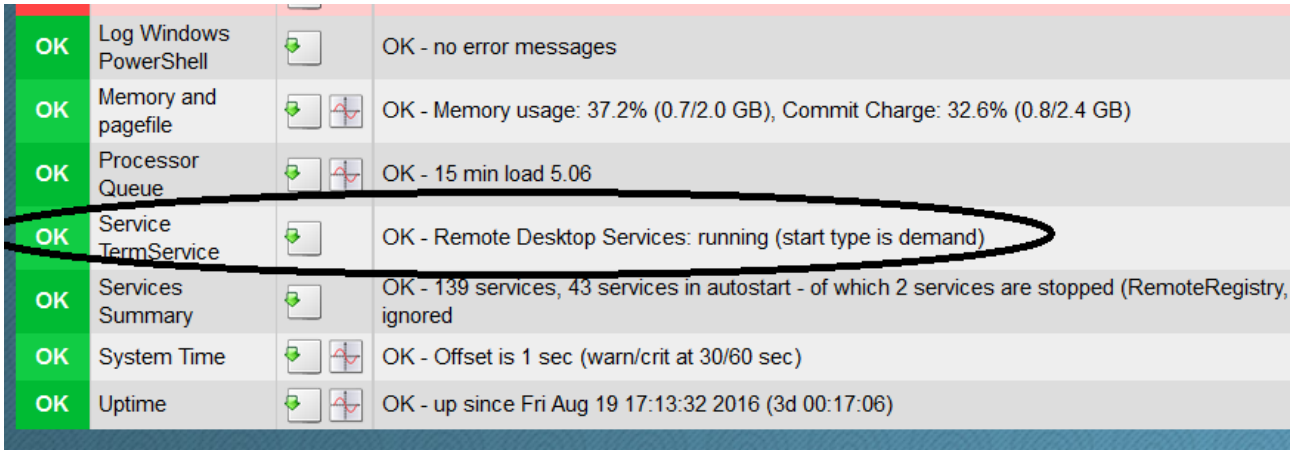


Create a rule like this one:



Force a host discovery using the command line or the GUI:



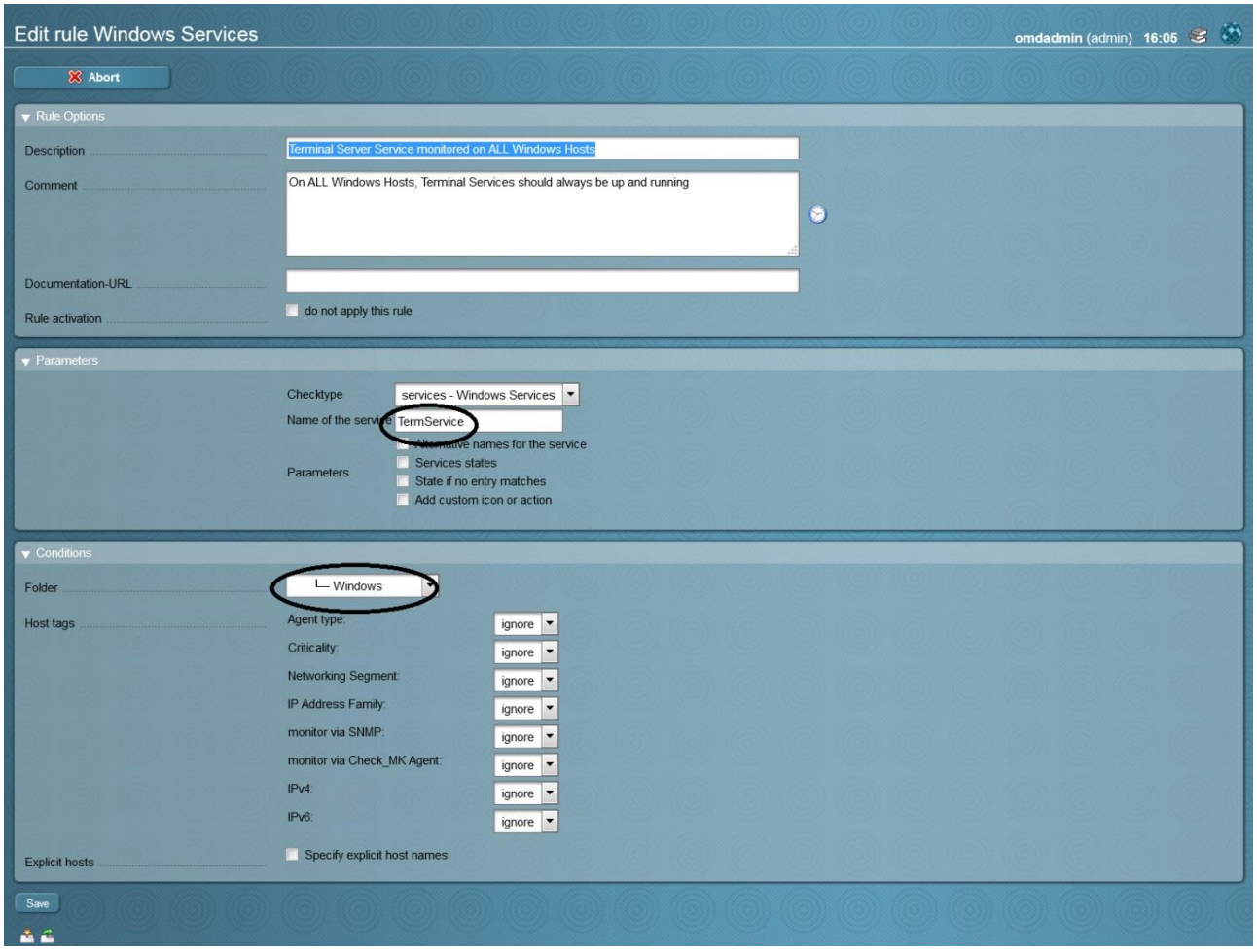


What WATO has done is write the following configuration file:

```
OMD[mysite]:/opt/omd/sites/mysite$ cat
/opt/omd/sites/mysite/etc/check_mk/conf.d/wato/windows/rules.mk

# Created by WATO
# encoding: utf-8


logwatch_rules = [
  ( [('C', u'testeventviewer*', u''), ('I', u'', u'')], ['/' + FOLDER_PATH + '/+'],
ALL_HOSTS, [u'Application$'], {'comment': u'This filter decides which events to take from
the "Application" Windows Event Log\n', 'description': u'Filter Application Windows Event
Log'} ),
] + logwatch_rules
```

```
static_checks.setdefault('services', [])

static_checks['services'] = [
  ( ('services', 'TermService', {}), ['/' + FOLDER_PATH + '/+'], ['w2012tst1'],
{'description': u'Terminal Server Service monitored on host w2012tst1'} ),
] + static_checks['services']


host_groups = [
  ( 'windowshg', ['/' + FOLDER_PATH + '/+'], ALL_HOSTS, {'comment': u'All hosts in
Windows folder are automatically placed in the windowshg hostgropu\n', 'description':
u'Windows hostgroup assignement'} ),
] + host_groups
```

If you have some services that should always be running on ALL windows hosts, the best way is create the rule that applies to ALL Windows Hosts; to do that, don't fill the *Explicit hosts* option.



This time, WATO changed the following configuration file:

```
OMD[mysite]:/opt/omd/sites/mysite$ tail -5
/opt/omd/sites/mysite/etc/check_mk/conf.d/wato/rules.mk

static_checks['services'] = [
```

```
  ( ('services', 'TermService', {}), [], ALL_HOSTS, {'comment': u'On ALL Windows Hosts,
Terminal Services should always be up and running\n', 'description': u'Terminal Server
Service monitored on ALL Windows Hosts'} ),
] + static_checks['services']
```

Let's add DnsCache (windows DNS client) to the monitored services, and do a test by stopping the DNSClient.

To add the new service, create another rule using WATO or manually change the configuration file and reload the configuration:

```
OMD[mysite]:/opt/omd/sites/mysite$ cat
/opt/omd/sites/mysite/etc/check_mk/conf.d/wato/windows/rules.mk

static_checks['services'] = [
  ( ('services', 'TermService', {}), ['/' + FOLDER_PATH + '/+'], ALL_HOSTS, {'comment':
u'On ALL Windows Hosts, Terminal Services should always be up and running\n',
'description': u'Terminal Server Service monitored on ALL Windows Hosts'} ),
  ( ('services', 'Dnscache', {}), ['/' + FOLDER_PATH + '/+'], ALL_HOSTS, {'comment':
u'The windows service that manage dnsclient should be always up and running\n',
'description': u'DnsCache monitored on ALL Windows Hosts'} ),
] + static_checks['services']
```

Now stop the service on the windows host and wait a minute. A *CRITICAL* service should be displayed!

Microsoft SQL Server

This is accomplished using the plugin “mssql.vbs” the documentation for which says the following:

```
The current implementation of the check uses the "trusted authentication" where no
user/password needs to be created in the MSSQL server instance by default. It is only
needed to grant the user as which the Check_MK windows agent service is running access to
the MSSQL database.

Another option is to create a mssql.ini file in MK_CONFDIR and write the credentials of a
database user to it which shall be used for monitoring:

[auth]
type = db
username = monitoring
password = secret-pw
```

I tested against Microsoft SQL Server 2014 64bit on Windows 2012 R2 using the default “trusted authentication”. This didn’t require any steps either on the SQL side or in check_mk.ini

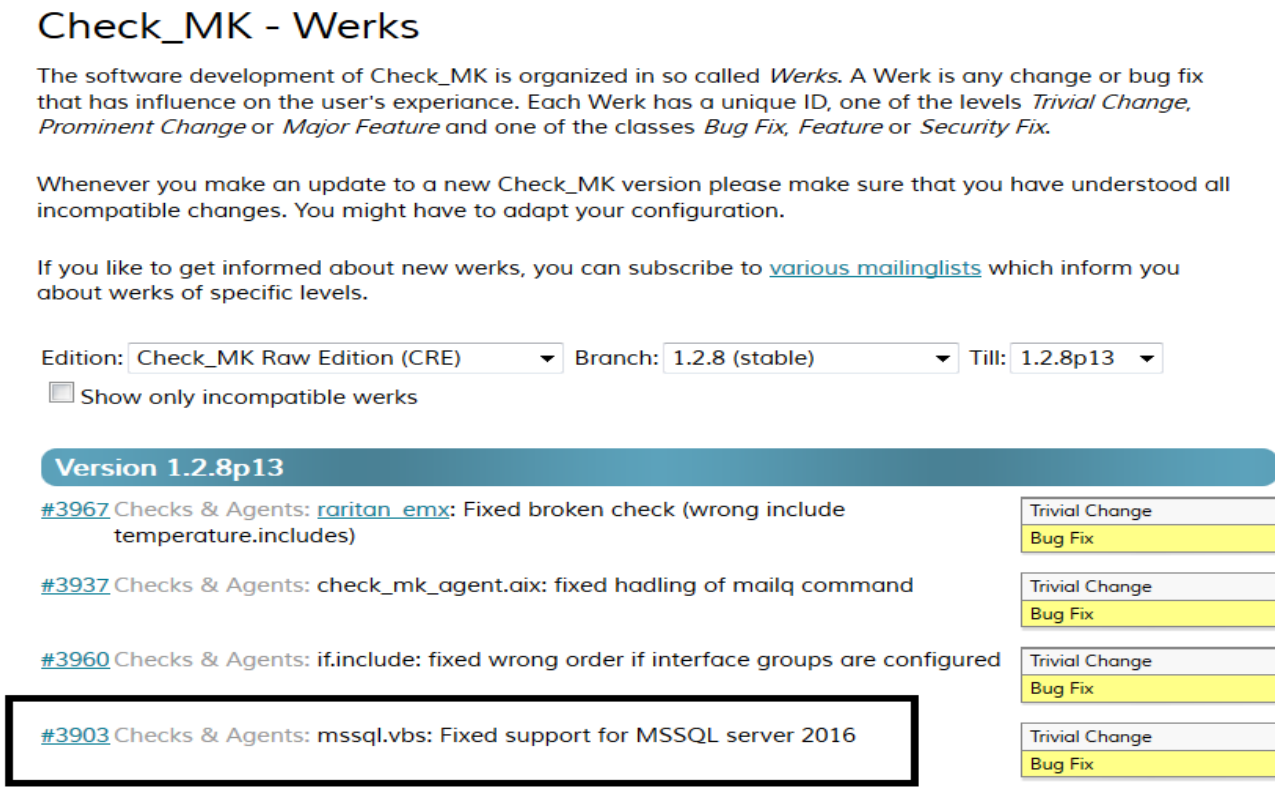
Steps:

- Copy mssql.vbs from check_mk host to the agent plugin folder, in my case: *C:\Program Files (x86)\check_mk\plugins*
- Restart the agent
- Do a service discovery adding unmonitored services
- Activate Changes

46 new services were added to my w2012tst1 host:

OK	SQLServer _Total File Sizes	OK - Data Files: 68.31 MB, Log files Used: 2.26 MB, Log Files total: 4.96 MB
OK	SQLServer _Total Transactions	OK - Transactions: 0.0/s, Tracked Transactions: 0.0/s, Write Transactions: 0.0/s
OK	SQLServer master File Sizes	OK - Data Files: 4.00 MB, Log files Used: 609.00 kB, Log Files total: 2.24 MB
OK	SQLServer master Transactions	OK - Transactions: 0.0/s, Tracked Transactions: 0.0/s, Write Transactions: 0.0/s
OK	SQLServer model File Sizes	OK - Data Files: 2.19 MB, Log files Used: 405.00 kB, Log Files total: 504.00 kB
OK	SQLServer model Transactions	OK - Transactions: 0.0/s, Tracked Transactions: 0.0/s, Write Transactions: 0.0/s
OK	SQLServer msdb File Sizes	OK - Data Files: 14.12 MB, Log files Used: 504.00 kB, Log Files total: 504.00 kB
OK	SQLServer msdb Transactions	OK - Transactions: 0.0/s, Tracked Transactions: 0.0/s, Write Transactions: 0.0/s
OK	SQLServer mssqlsystemresource File Sizes	OK - Data Files: 40.00 MB, Log files Used: 438.00 kB, Log Files total: 1.24 MB
OK	SQLServer mssqlsystemresource Transactions	OK - Transactions: 0.0/s, Tracked Transactions: 0.0/s, Write Transactions: 0.0/s
OK	SQLServer tempdb File Sizes	OK - Data Files: 8.00 MB, Log files Used: 355.00 kB, Log Files total: 504.00 kB
OK	SQLServer tempdb Transactions	OK - Transactions: 0.0/s, Tracked Transactions: 0.0/s, Write Transactions: 0.0/s
OK	SQLServer:Buffer_Manager None buffer_cache_hit_ratio	OK - 100%
OK	SQLServer:Catalog_Metadata _Total cache_hit_ratio	OK - 66%
OK	SQLServer:Catalog_Metadata master cache_hit_ratio	OK - 72%
OK	SQLServer:Catalog_Metadata model cache_hit_ratio	OK - 0%
OK	SQLServer:Catalog_Metadata msdb cache_hit_ratio	OK - 28%
OK	SQLServer:Catalog_Metadata mssqlsystemresource cache_hit_ratio	OK - 74%
OK	SQLServer:Catalog_Metadata tempdb cache_hit_ratio	OK - 60%
OK	SQLServer:Locks _Total Locks	OK - Requests: 5.9/s, Timeouts: 0.0/s, Deadlocks: 0.0/s, Waits: 0.0/s

Note: Since version 1.2.8p13, Microsoft Sql Server 2016 is also supported



Check_MK - Werks

The software development of Check_MK is organized in so called *Werks*. A Werk is any change or bug fix that has influence on the user's experiance. Each Werk has a unique ID, one of the levels *Trivial Change*, *Prominent Change* or *Major Feature* and one of the classes *Bug Fix*, *Feature* or *Security Fix*.

Whenever you make an update to a new Check_MK version please make sure that you have understood all incompatible changes. You might have to adapt your configuration.

If you like to get informed about new werks, you can subscribe to various mailinglists which inform you about werks of specific levels.

Edition: Check_MK Raw Edition (CRE) Branch: 1.2.8 (stable) Till: 1.2.8p13

Show only incompatible werks

Version 1.2.8p13

#3967 Checks & Agents: raritan_emx: Fixed broken check (wrong include temperature.includes) — Trivial Change, Bug Fix

#3937 Checks & Agents: check_mk_agent.aix: fixed hadling of mailq command — Trivial Change, Bug Fix

#3960 Checks & Agents: if.include: fixed wrong order if interface groups are configured — Trivial Change, Bug Fix

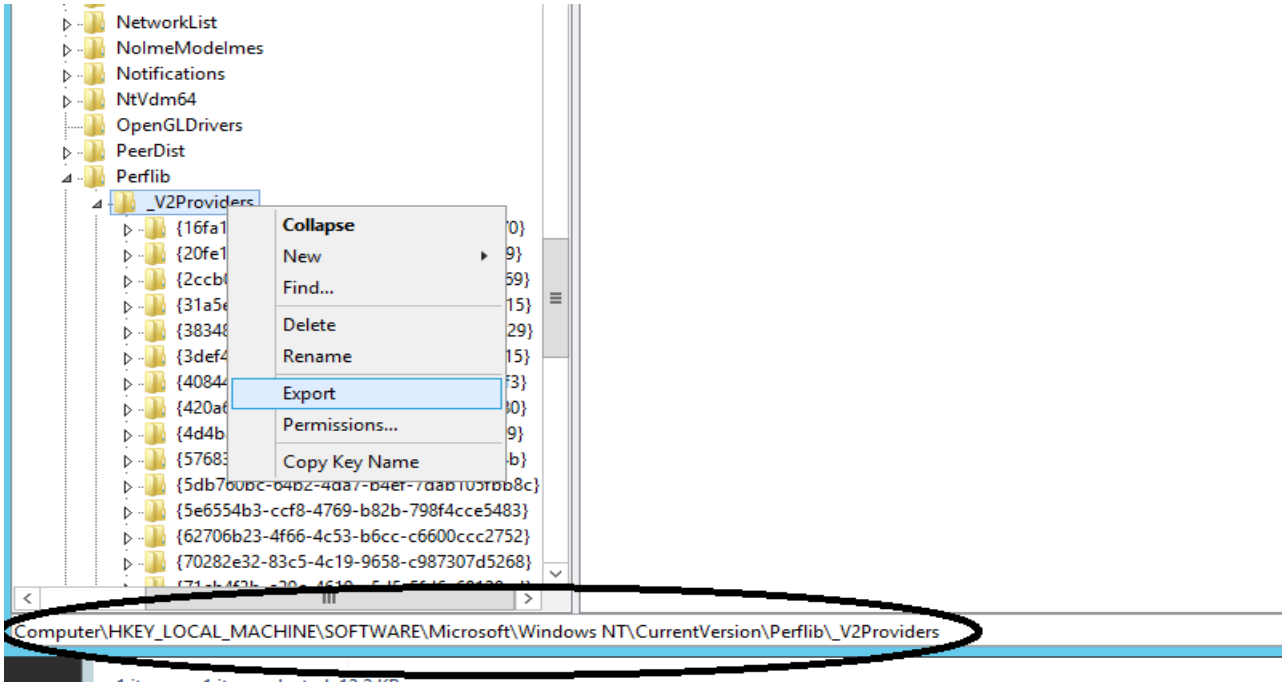
#3903 Checks & Agents: mssql.vbs: Fixed support for MSSQL server 2016 — Trivial Change, Bug Fix

Microsoft Terminal Services

Several Windows checks are based on *Performance Counters*. These are special objects provided by the Windows operating system that contain information about throughput, queue lengths, latencies and other numbers of the system and applications like MS Exchange, MSSQL, IIS etc
Because there is no native support for Terminal Services, we need to take advantage of Performance Counters. I fought a little bit with this task but, thanks again to the mailing list, I was able to do it in this way:

- On the windows host run *regedit* and export the following key:

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib_V2Providers]



- Open the file and search the string *Terminal Services*

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Perflib\_V2Providers\{f3b975e7-e068-4f66-81ef-b23e0a0e64c9}\{fc9e399c-
c70a-4458-8430-ca249c371eb3}]
"NameResource"=dword:00000001
"ExplainResource"=dword:00000003
"NeutralName"="Terminal Services"
"InstanceType"=dword:00000000
"First Counter"=dword:00000780
"Last Counter"=dword:00000786
"CounterBlock"=hex:01,00,00,00,00,00,01,00,01,00,00,00,00,00,00,00,64,00,00,00,\
  00,00,00,00,05,00,00,00,07,00,00,00,00,00,00,00,ff,ff,ff,ff,ff,ff,ff,ff,ff,\
.....
```

- Take the hexdecimal value of *"First Counter"* and convert in decimal. In this case:

00000780 = 1920

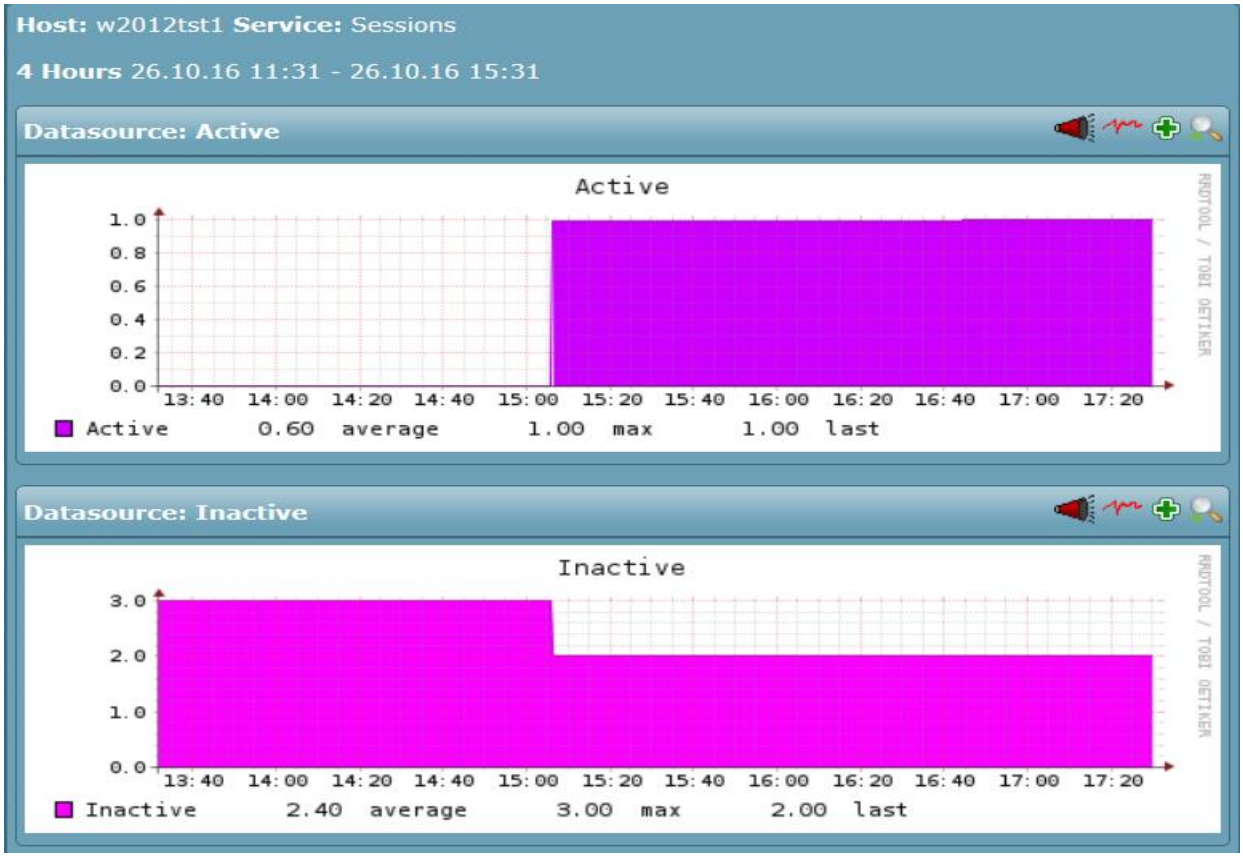
- Edit *check_mk.ini* and add the following string in the *winperf* section

```
[winperf]
    # Select counters to extract. The following counters
    # are needed by checks shipped with check_mk.
    # counters = 10332:msx_queues
    # counters = 638:tcp_conn
    counters = 1920:ts_sessions
```

- Restart the agent

- Do a service discovery adding unmonitored services

OK	Processor Queue		OK - 15 min load 5.54
OK	Service Dnscache		OK - DNS Client: running (start type is auto)
OK	Service TermService		OK - Remote Desktop Services: running (start type is
OK	Services Summary		OK - 156 services, 53 services in autostart - of which 3 sppsvc, TrustedInstaller), 0 services stopped but ignor
OK	Sessions		OK - 1 Active, 2 Inactive
OK	SQLServer _Total File Sizes		OK - Data Files: 68.31 MB, Log files Used: 2.26 MB, L
OK	SQLServer _Total Transactions		OK - Transactions: 0.0/s, Tracked Transactions: 0.0/s,
OK	SQLServer master File Sizes		OK - Data Files: 4.00 MB, Log files Used: 609.00 kB, L
OK	SQLServer master Transactions		OK - Transactions: 0.0/s, Tracked Transactions: 0.0/s

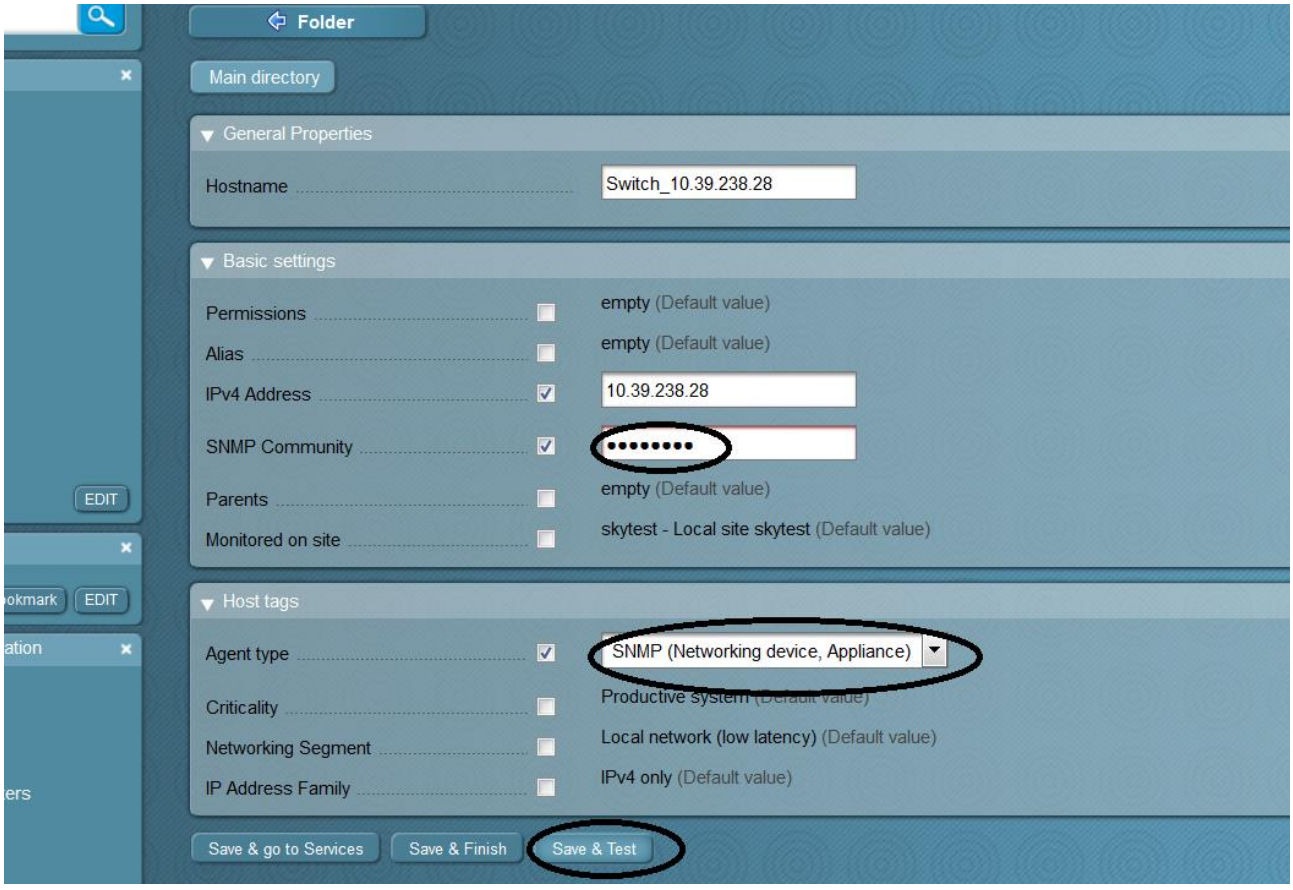


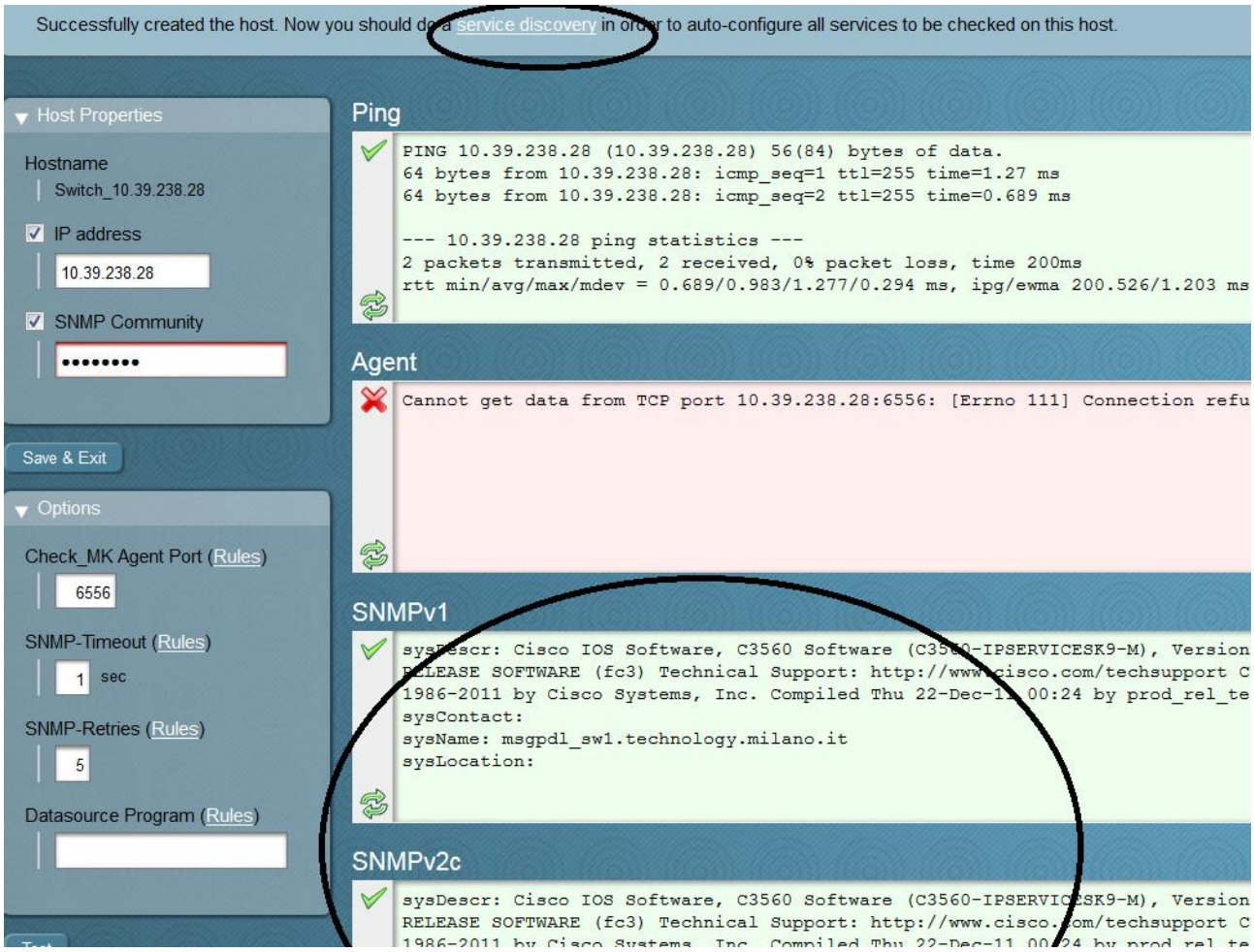
Network Devices

Network devices (switches, router, firewall, balancer etc.) are monitored using the SNMP protocol. SNMP uses UDP as its transport protocol. If management traffic needs to traverse firewalls, make sure that the following default ports are open:

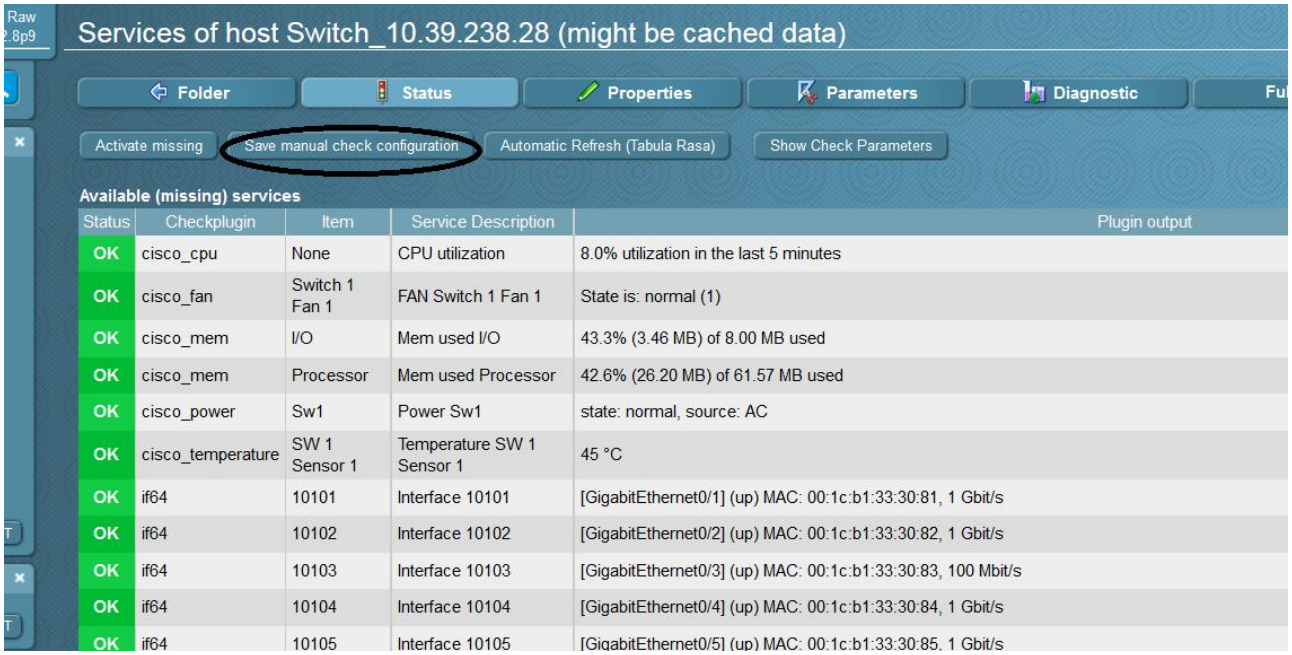
- UDP 161: Used when management stations communicate with agents, e.g. Polling
- UDP 162: Used when agents send unsolicited Traps to the management station

During the wizard, please be sure to select *SNMP (Networking device, Appliance)* in the Agent type combo box.

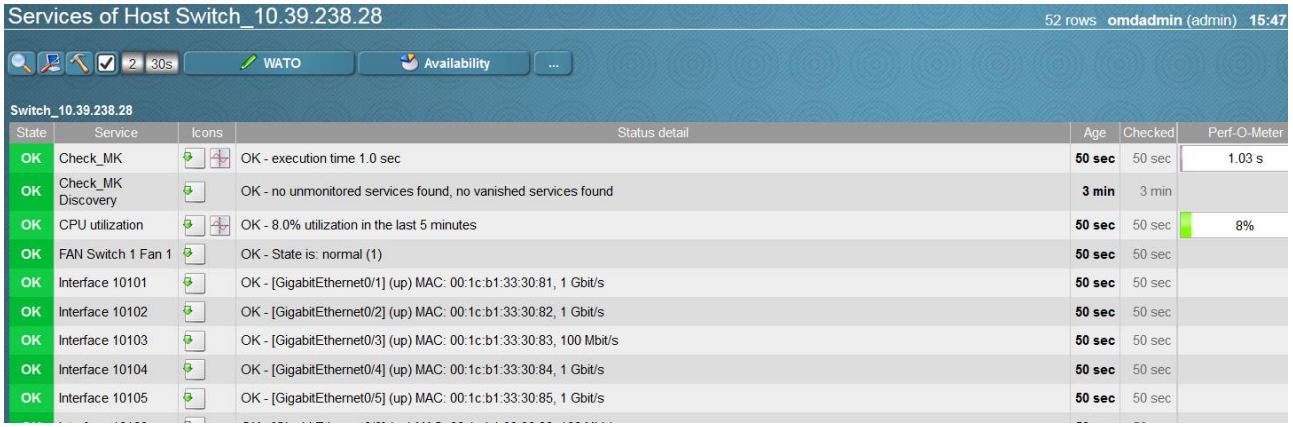




Click on *Service Discover, Save manual check configuration*



As usual apply the changes and wait a while to have the new device appear.



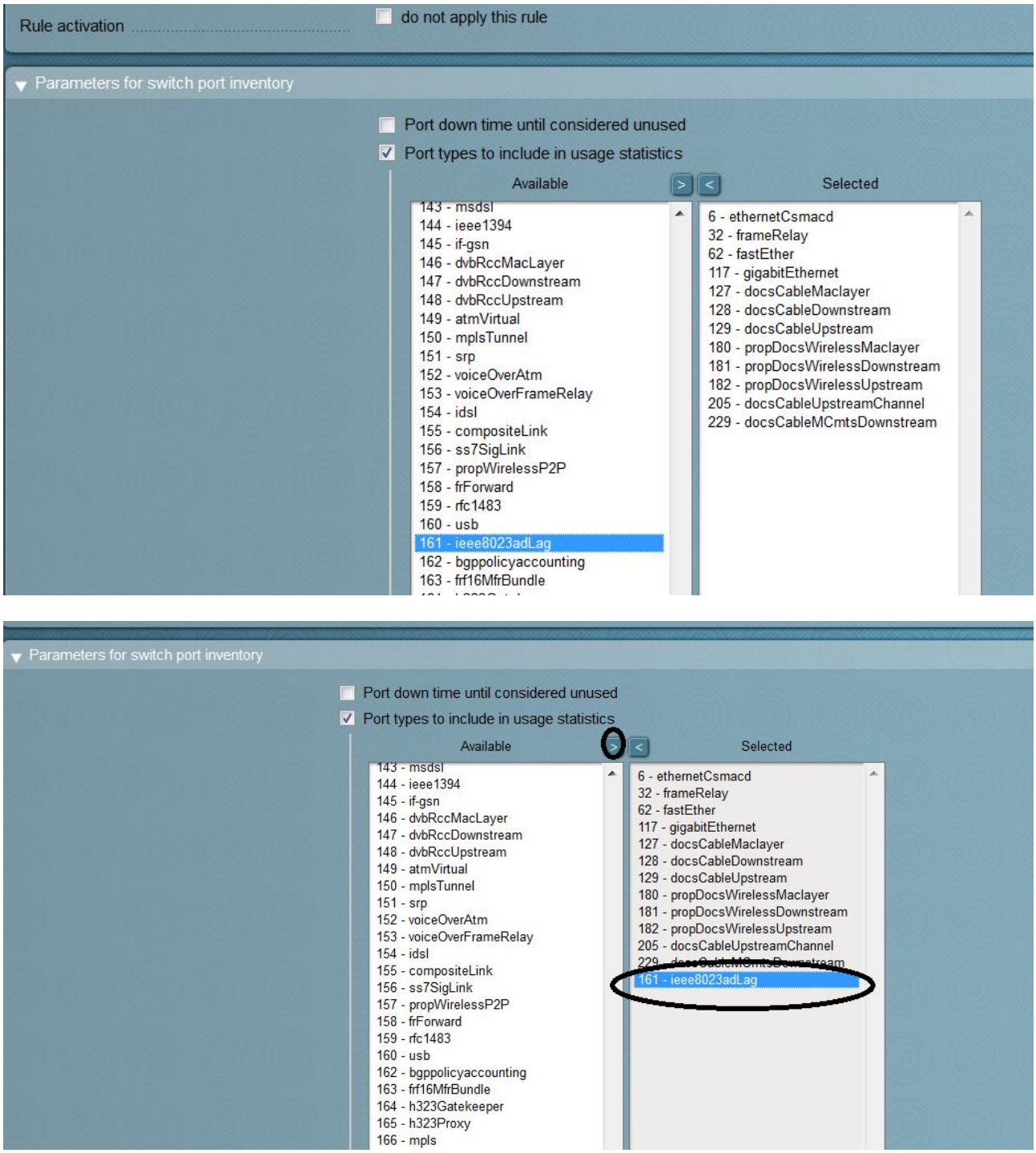
In this case a *CRITICAL* service will fire up in case of hardware failure and, depending on the check parameters, *WARN* or *CRIT* when the port status changes (i.e. is down), when the link speed changes (e.g. a port expected to be set to 1GBit/s operates only at 100MBit/s), when the absolute or percentage traffic of a port exceeds certain levels or if the rate of errors or discards exceeds configurable limits.
By default, Check_MK doesn't inventory Port-Channels. Port-Channels are aggregated physical interfaces which are usually used for inter-switch connectivity. After a Google search, I found a post explaining how to fix that: https://sitweak.wordpress.com/2012/08/16/monitoring-port-channel-on-cisco-switchesrouters-with-check_mk/

I don't understand the reason behind that choice - in my opinion the default should be to always monitor.


Rule-Based Configuration of Host & Service Parameters
No Changes
Main Menu
Used Rulesets
Ineffective rules
Dep
Main directory
Search for rule sets:
Search
Active checks (HTTP, TCP, etc.)
Configure active networking checks like HTTP and TCP
Grouping
Assignment of host & services to host, service and contacts groups.
Access to Agents
Settings concerning the connection to the Check_MK and SNMP agents
Parameters for discovered services
Levels and other parameters for checks found by the Check_MK service discovery.
Hardware/Software-Inventory
Configuration of the Check_MK Hardware and Software Inventory System
Event Console
Settings and Checks dealing with the Check_MK Event Console


Hardware/Software-Inventory
No Changes
Main Menu
All Rulesets
Folder
Main directory
Hardware/Software-Inventory
Do hardware/software Inventory 1
Export List of Software packages as CSV file 0
Parameters for switch port inventory 0


Raw 2.8p9
Parameters for switch port inventory
No Changes
Main Menu
Hardware/Software-In...
Used Rulesets
Main directory
Matching: Each parameter is defined by the first matching rule where that parameter is set (checked).
There are no rules defined in this set.
Create rule in folder:
Main directory

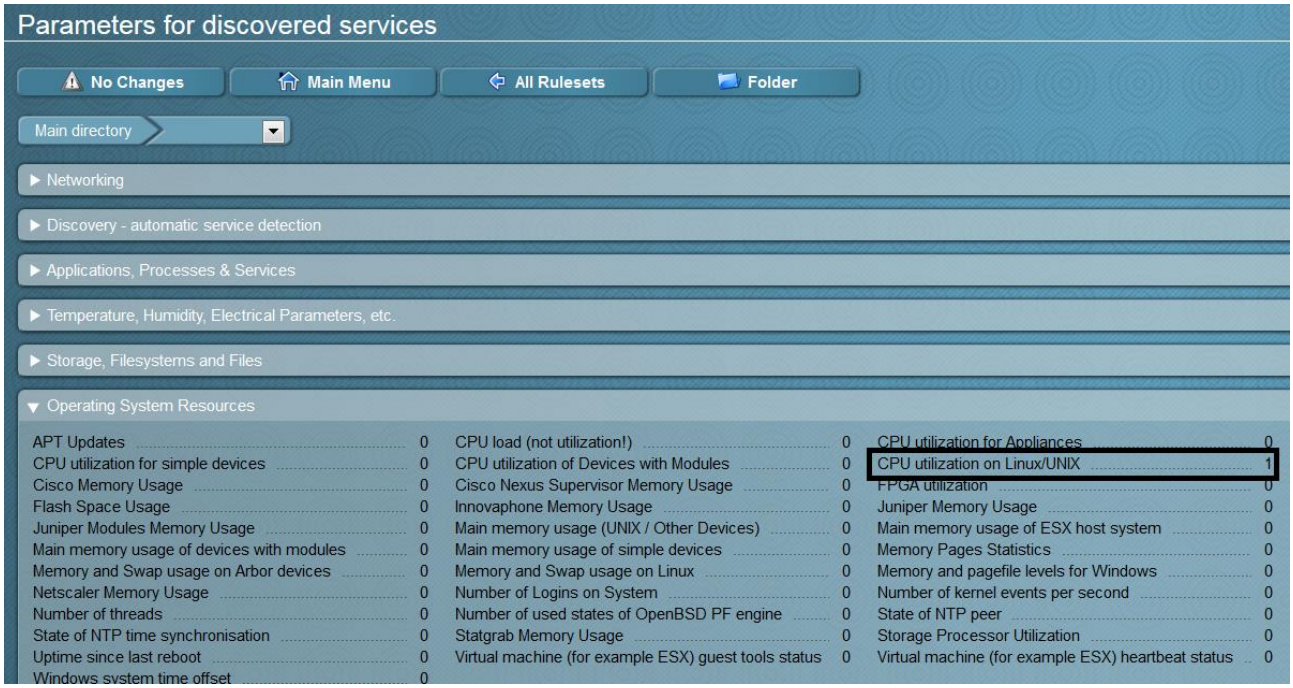


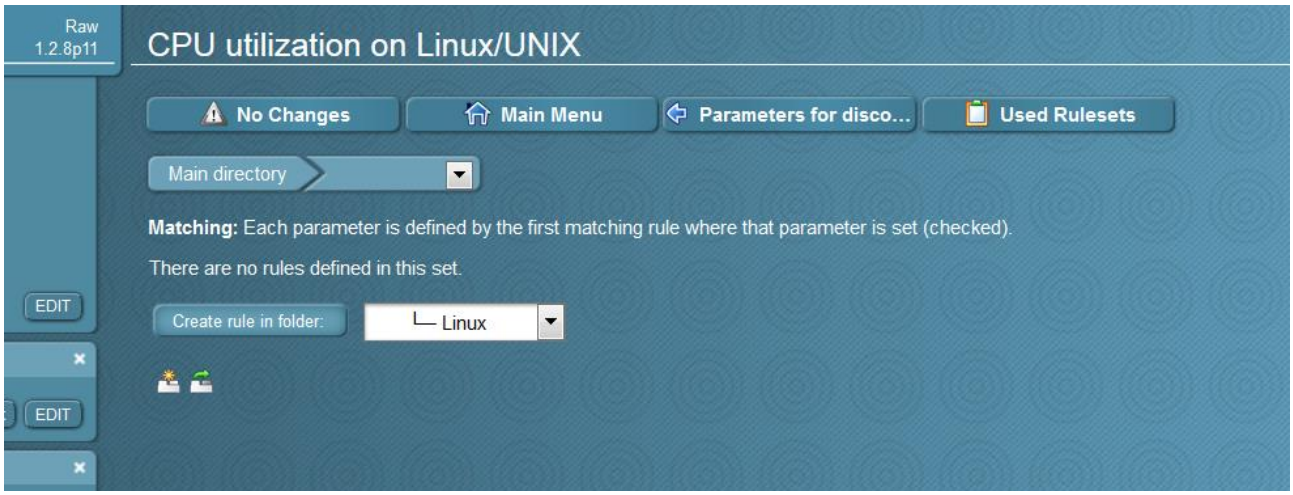
Click Save and Activate changes

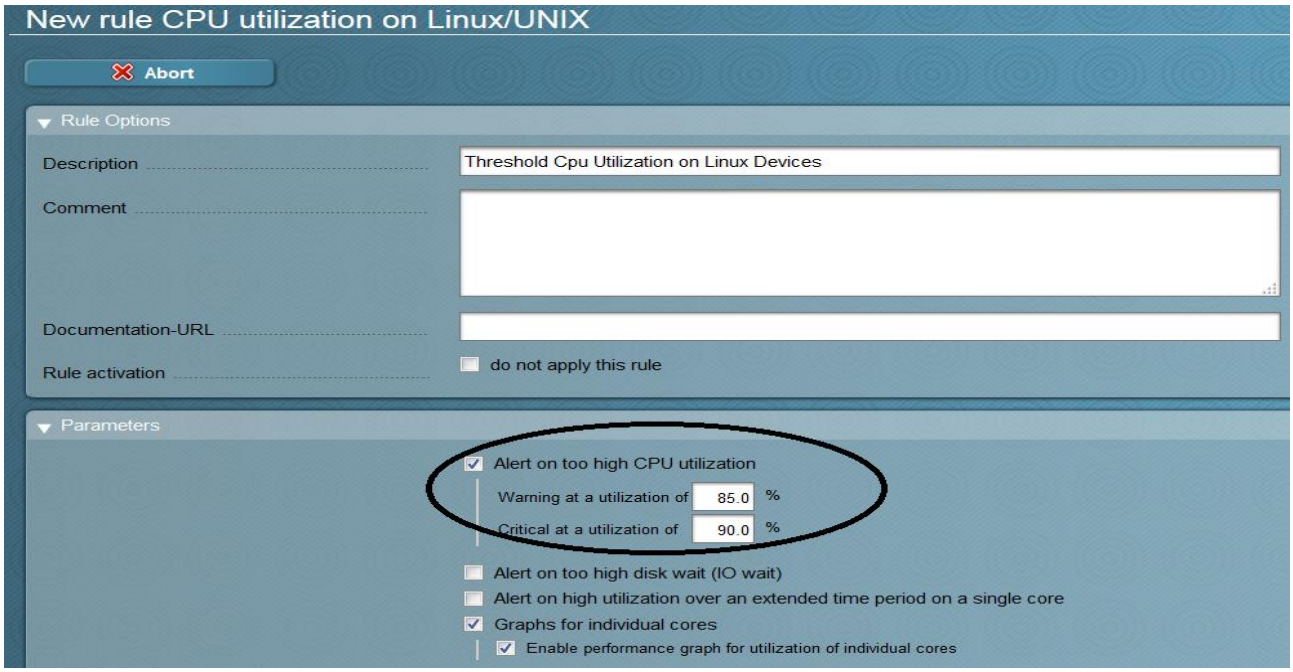
Managing Thresholds

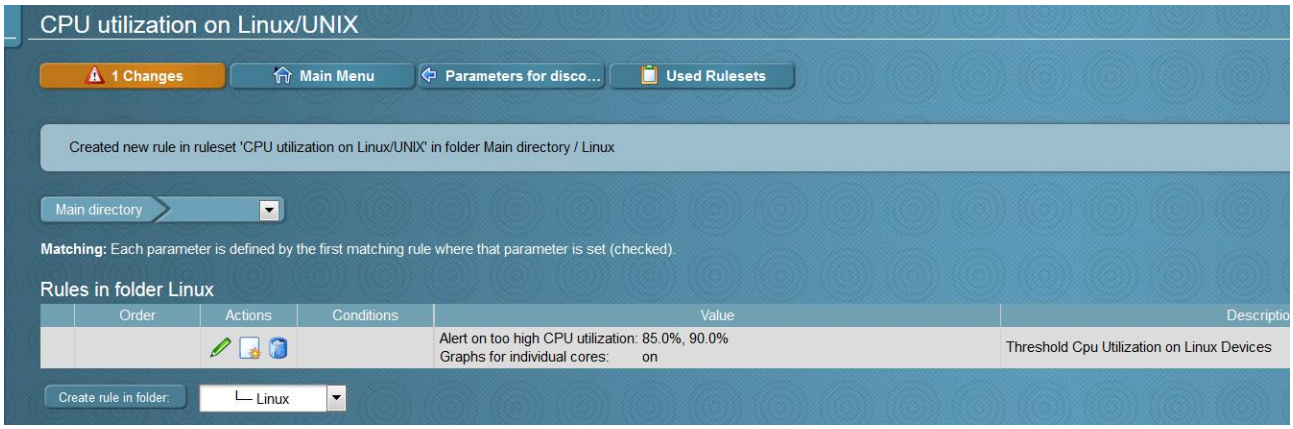
A threshold is a range with an alert level, either warning or critical. The theory is that the plugin will do some sort of check which returns back a numerical value, or metric, which is then compared to the warning and critical thresholds. To avoid useless alerts, I suggest to define a certain number of *check attempts* before to send out alarms and notifications. For example: CPU spikes are quite usual and normal thus it would be useful to be notified only when its consumption is too much high for more than a specified time period.
In this example, a CPU threshold is setup so that a *CRITICAL* service will be created only if the percentage of CPU utilization is above 90 % for more than 5 minutes.

WATO, Host & Service Parameters, Parameters for discovered services, CPU utilization on Linux/UNIX



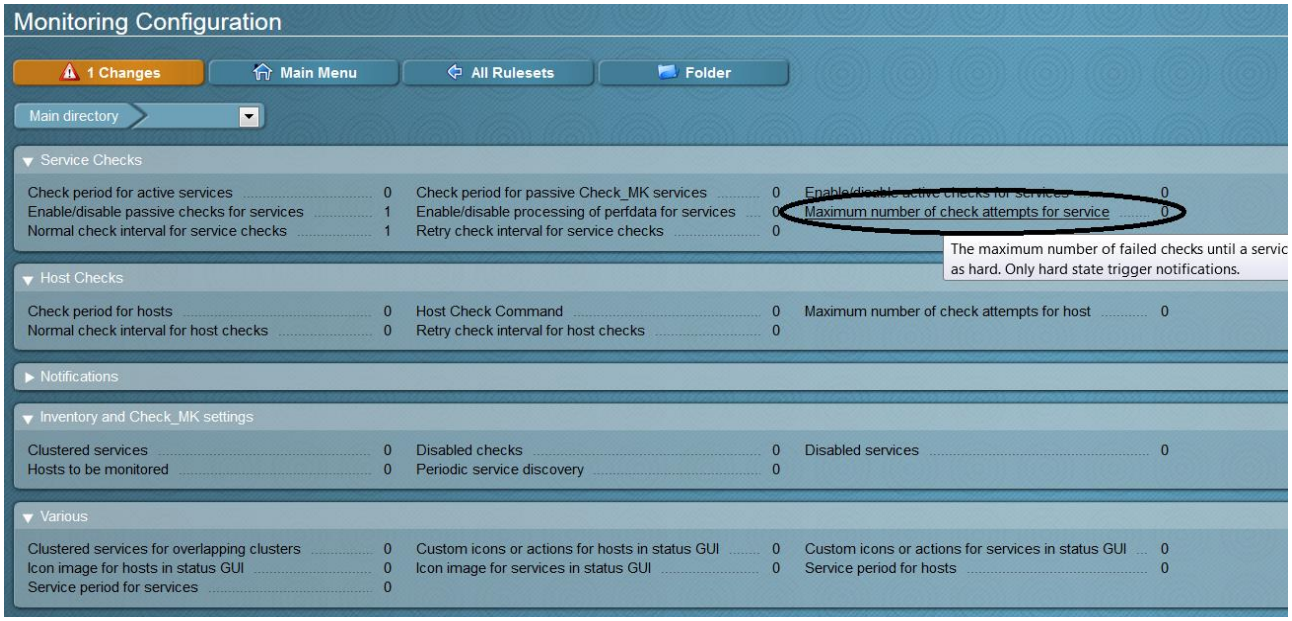




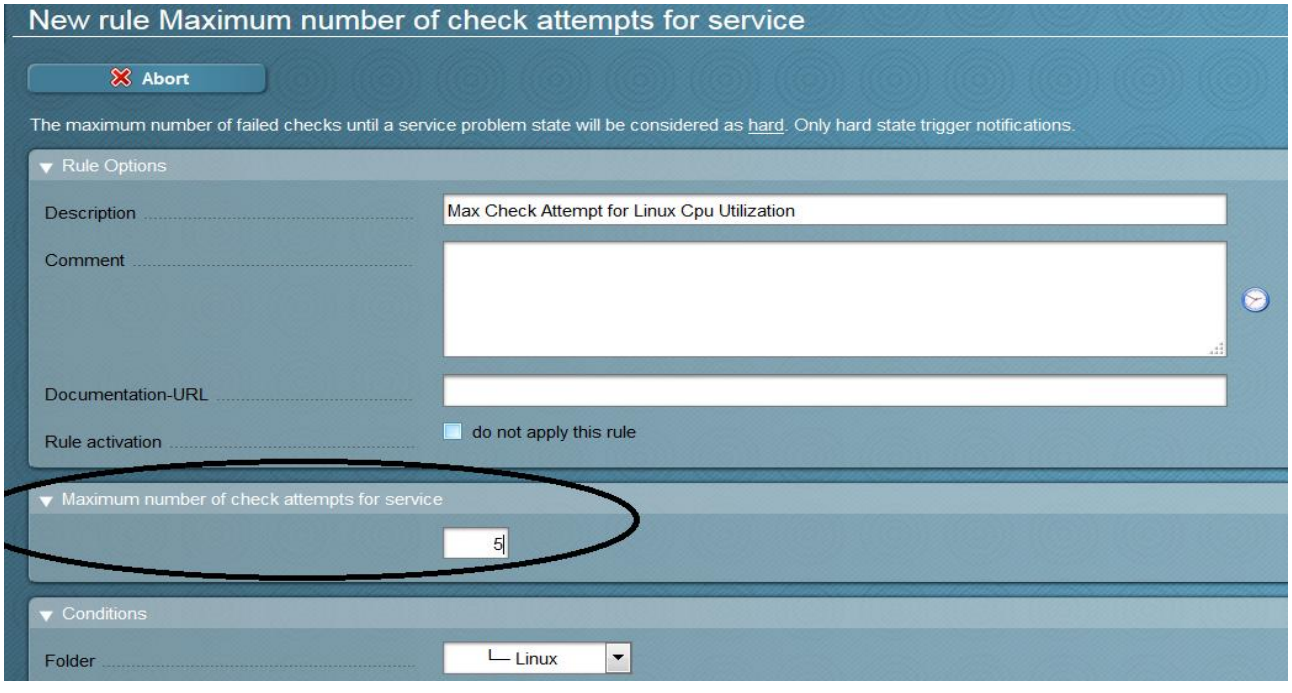


Given I'm using the default check periods of 60 seconds all I need to do is set *max_check_attempt* to 5 (60*5=300 seconds), which gives five minutes of checks before the state switches from *SOFT* to *HARD*.

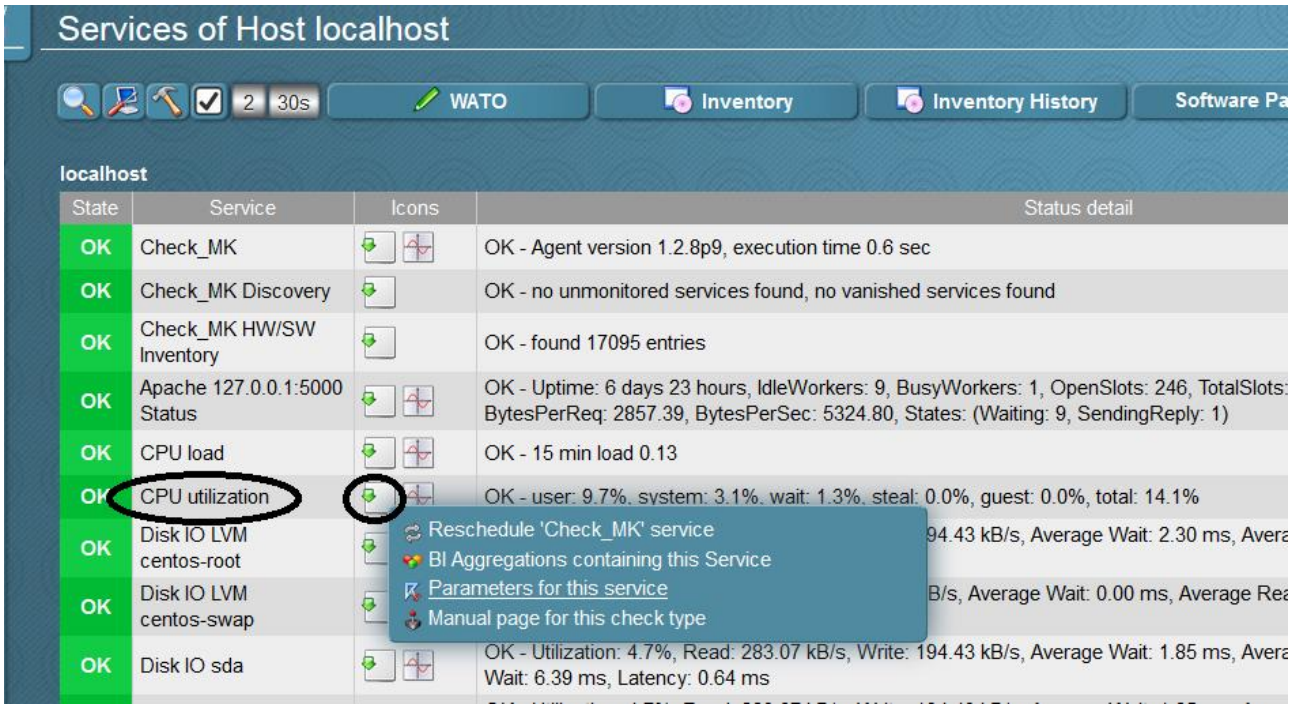
WATO, Host & Service Parameters, Monitor Configuration, Maximum number of check attempts for service

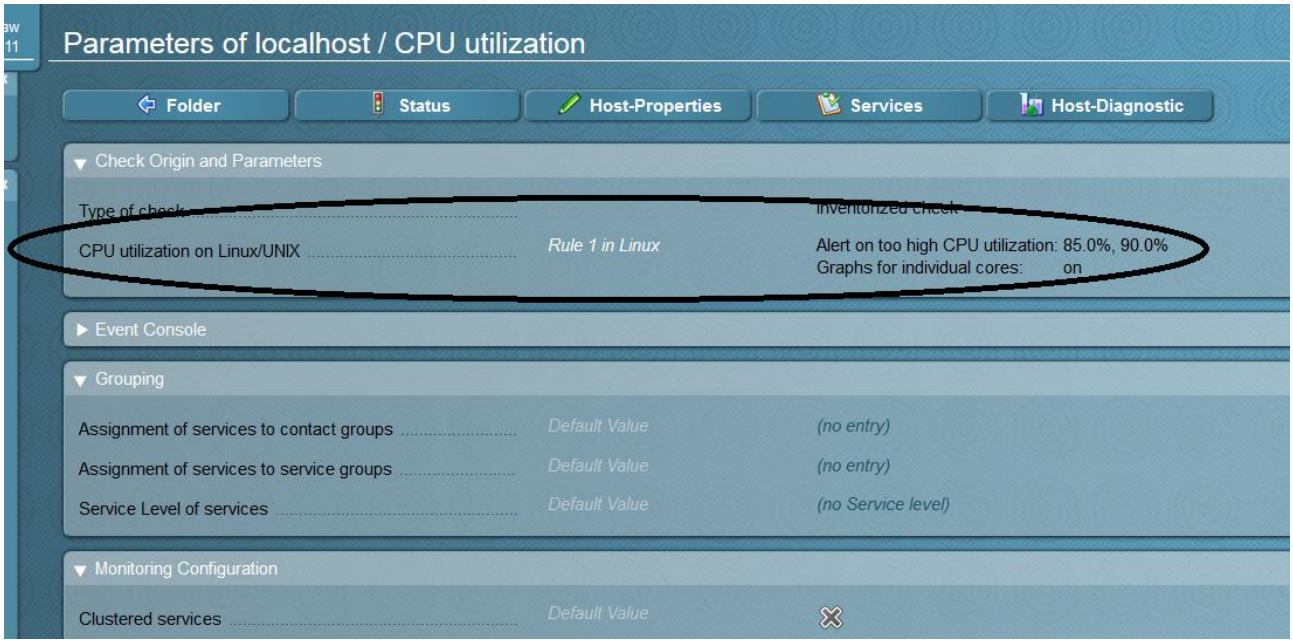

Monitoring Configuration
1 Changes
Main Menu
All Rulesets
Folder
Main directory
Service Checks
Check period for active services 0
Enable/disable passive checks for services 1
Normal check interval for service checks 1
Check period for passive Check_MK services 0
Enable/disable processing of perfdata for services 0
Retry check interval for service checks 0
Maximum number of check attempts for service 0
The maximum number of failed checks until a servic
as hard. Only hard state trigger notifications.
Host Checks
Check period for hosts 0
Normal check interval for host checks 0
Host Check Command 0
Retry check interval for host checks 0
Maximum number of check attempts for host 0
Notifications
Inventory and Check_MK settings
Clustered services 0
Hosts to be monitored 0
Disabled checks 0
Periodic service discovery 0
Disabled services 0
Various
Clustered services for overlapping clusters 0
Icon image for hosts in status GUI 0
Service period for services 0
Custom icons or actions for hosts in status GUI 0
Icon image for services in status GUI 0
Custom icons or actions for services in status GUI 0
Service period for hosts 0


Maximum number of check attempts for service
1 Changes
Main Menu
Monitoring Configura...
Used Rulesets
Main directory
Matching: The first matching rule defines the parameter.
There are no rules defined in this set.
Create rule in folder:
Linux


New rule Maximum number of check attempts for service
Abort
The maximum number of failed checks until a service problem state will be considered as hard. Only hard state trigger notifications.
Rule Options
Description
Max Check Attempt for Linux Cpu Utilization
Comment
Documentation-URL
Rule activation
do not apply this rule
Maximum number of check attempts for service
5
Conditions
Folder
Linux

To check if the rule has been applied, an easy way is to choose a server and look for the “CPU Utilization” service parameters

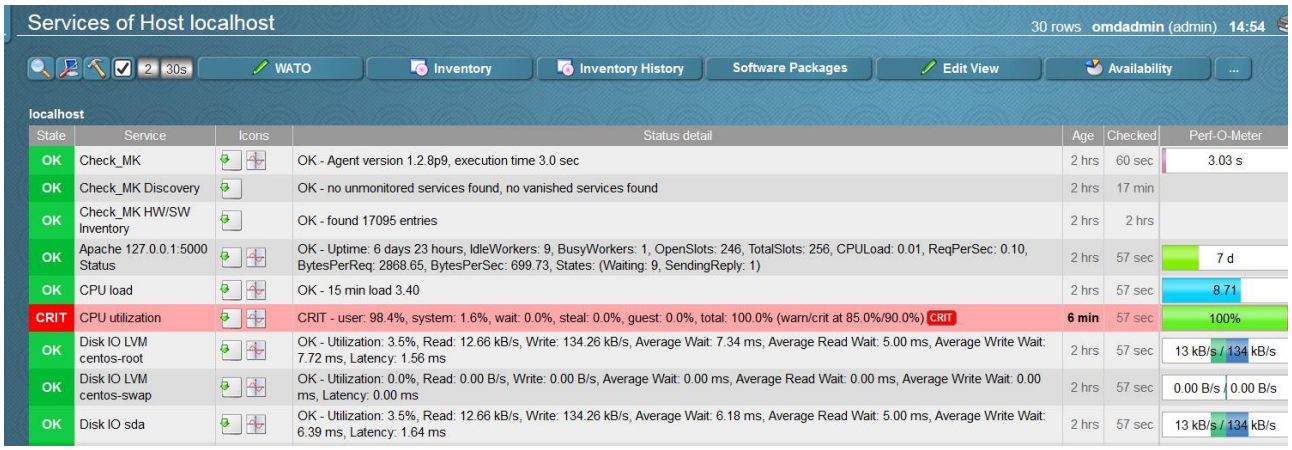




Let’s do some testing using the *stress* utility

```
[root@checkmktst1 ~]# stress --cpu 8 --timeout 600
stress: info: [12082] dispatching hogs: 8 cpu, 0 io, 0 vm, 0 hdd
```

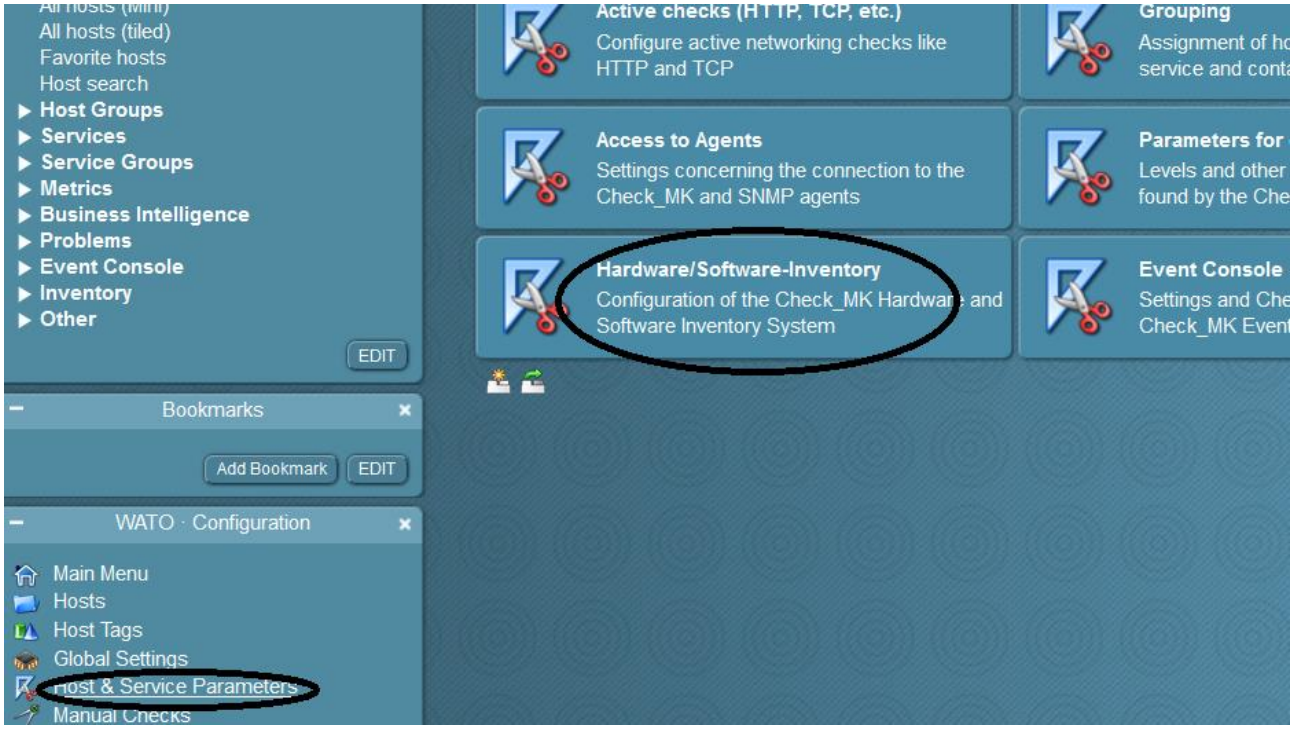
After 5 minutes, the service should be in CRITICAL state



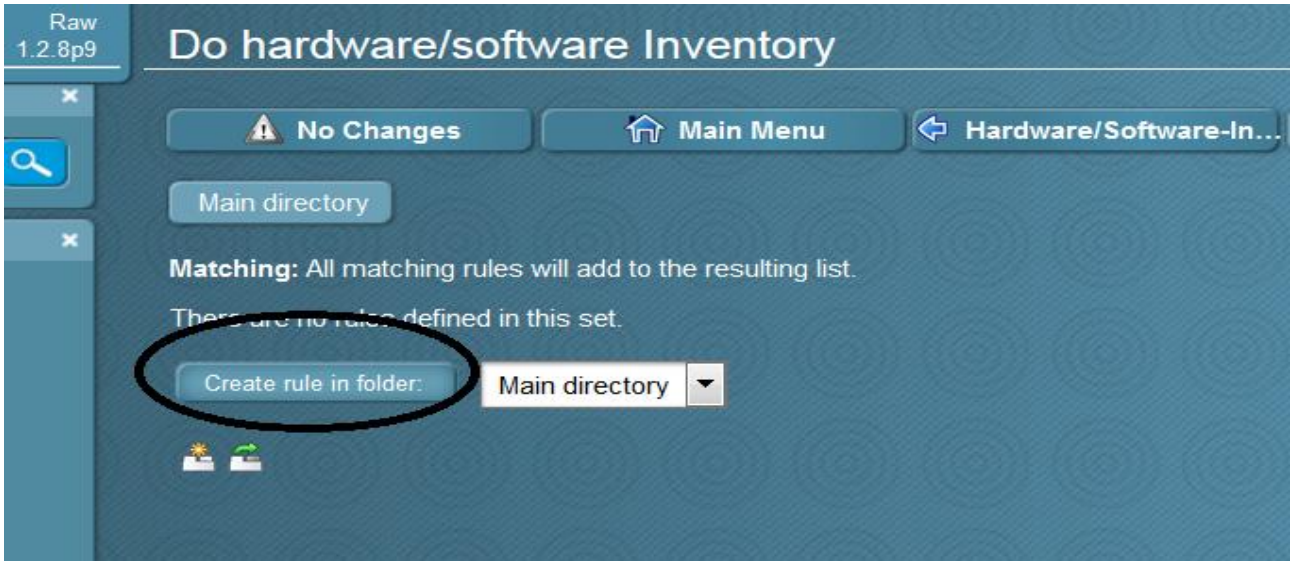
Hardware & Software Inventory

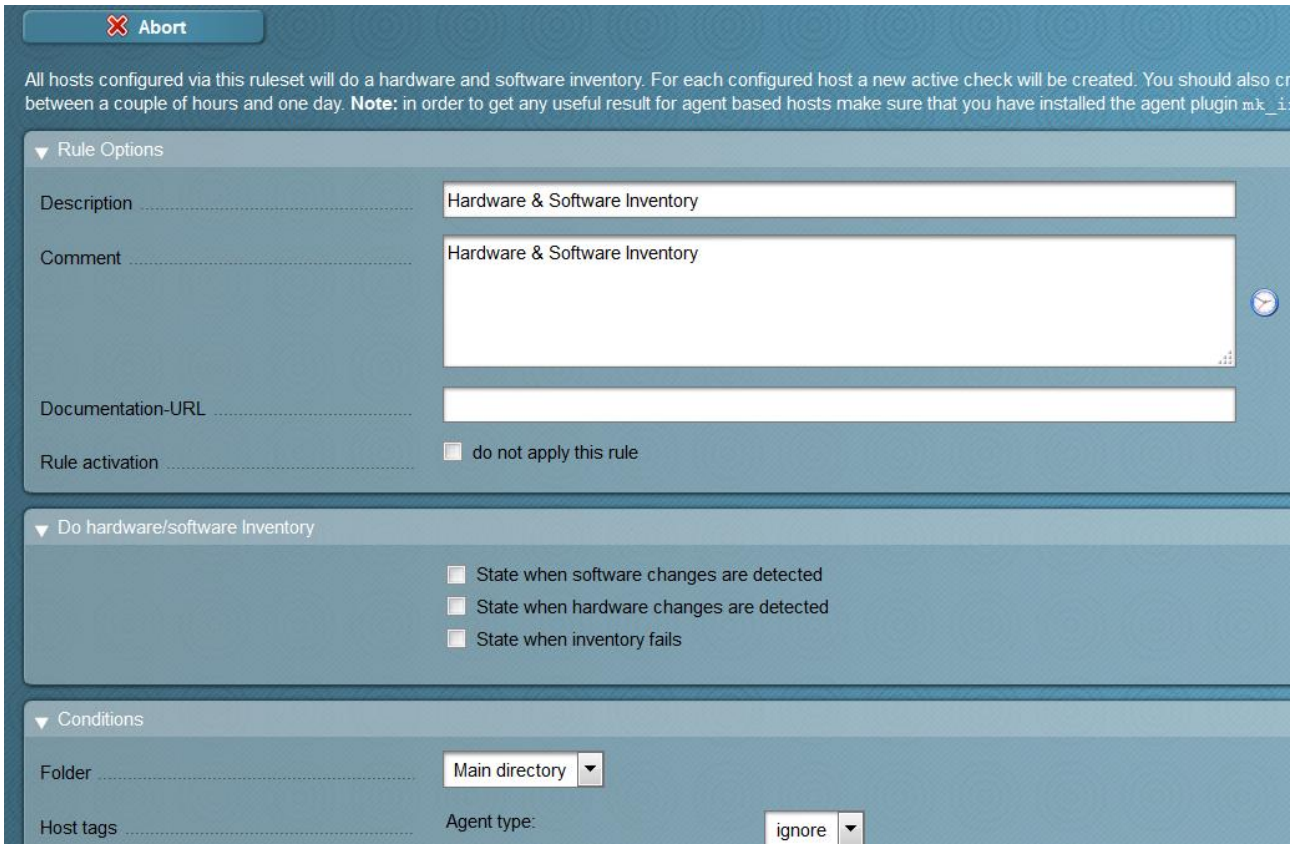
Check_MK supports hardware & software inventories. While SNMP devices don't require any additional components, for Windows & Linux devices we need a plugin.

The first step is to enable *Hardware/Software-Inventory* by creating a rule:




Check MK
Raw 1.2.8p9
Quicksearch
Views
Overview
Hosts
All hosts
All hosts (Mini)
All hosts (tiled)
Favorite hosts
Host search
Host Groups
Services
Service Groups
Metrics
Business Intelligence
Problems
Hardware/Software-Inventory
No Changes
Main Menu
All Rulesets
Main directory
Hardware/Software-Inventory
Do hardware/software Inventory
0
Export List of Software packages as


Raw 1.2.8p9
Do hardware/software Inventory
No Changes
Main Menu
Hardware/Software-In...
Main directory
Matching: All matching rules will add to the resulting list.
defined in this set.
Create rule in folder:
Main directory



Click *Save* and remember to apply changes.

Now it's time to install the plugin for both Linux and Windows server:

Linux:

-Copy the "mk_inventory" plugin in the "local" folder of the linux agent. In my case the path is:

/usr/lib/check_mk_agent/local/mk_inventory

Make sure it is executable

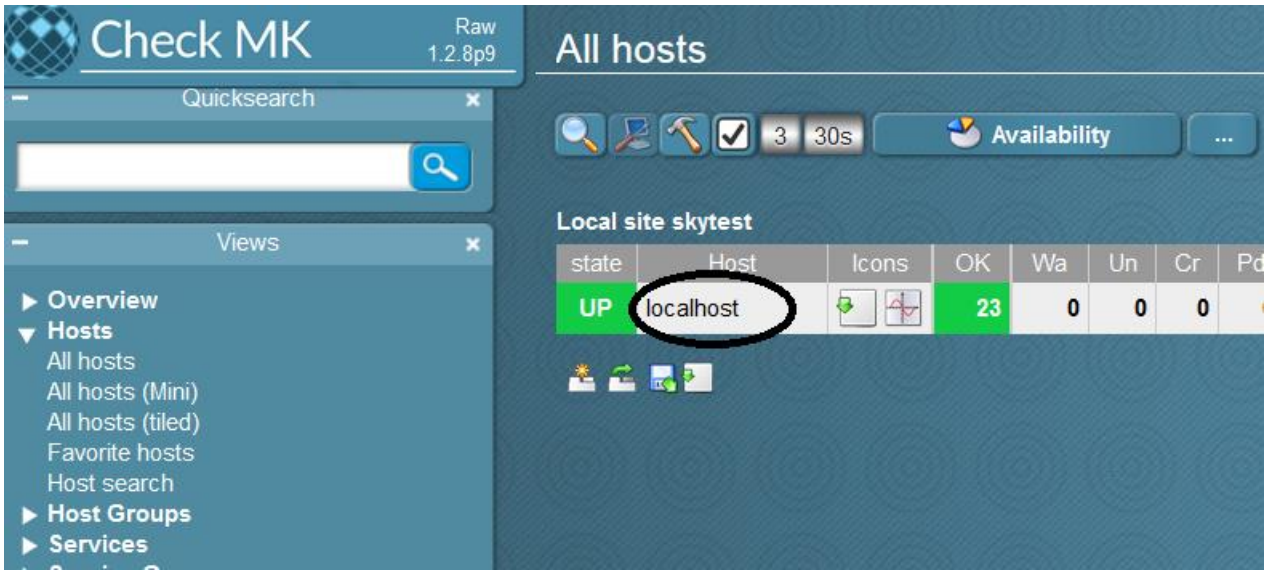
```
chmod +x /usr/lib/check_mk_agent/local/mk_inventory
```

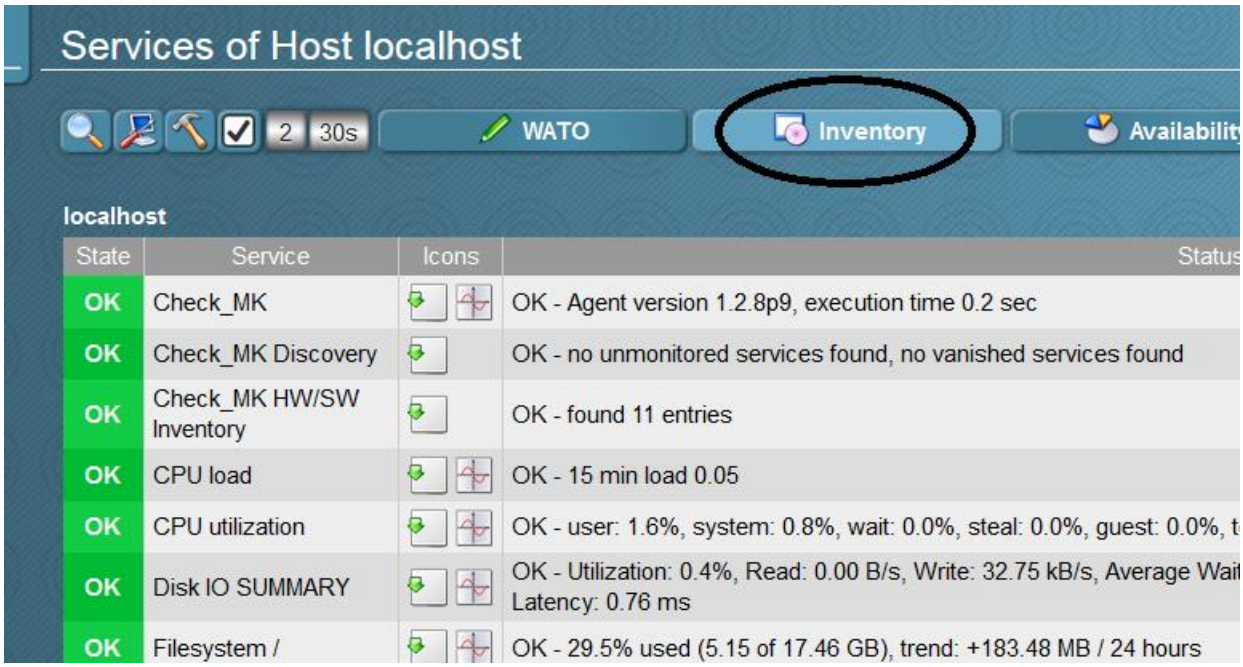
If you are not sure about it, you can check it by simply running the agent from the command line and checking the output which should show the current configuration:

OMD[mysite]:~$ **/usr/bin/check_mk_agent**
<<<check_mk>>>
Version: 1.2.8p9
AgentOS: linux
Hostname: checkmktst1
AgentDirectory: /etc/check_mk
DataDirectory: /var/lib/check_mk_agent
SpoolDirectory: /var/lib/check_mk_agent/spool
PluginsDirectory: /usr/lib/check_mk_agent/plugins
LocalDirectory: /usr/lib/check_mk_agent/local
……

-Force inventory on check_mk server:

OMD[mysite]:~$ **cmk -i**





Hostname	localhost

Inventory Tree

- Hardware
  - BIOS
  - Chassis
  - Processor
  - Memory (RAM)
  - System

Manufacturer	VMware, Inc.
Product	VMware Virtual Platform
Serial Number	VMware-56 4d fe 15 f2 57 70 68-90 2a 67 09 86 10 91 d2
Uuid	564DFE15-F257-7068-902A-6709861091D2
Version	None

- Networking

Hostname	checkmktst1

  - IP Addresses
  - Routes
- Software
  - Operating System

Code Name	Core
Kernel Architecture	x86_64
Kernel Version	3.10.0-327.28.2.el7.x86_64
Name	CentOS Linux release 7.2.1511
Type	linux
Version	7.2.1511

- Packages

Ope

Name	Version	Architecture	Type	Description
ebtables	2.0.10	x86_64	rpm	Ethernet Bridge frame table administration tool
openssh	6.6.1p1	x86_64	rpm	An open source implementation of SSH protocol versions 1 and 2
hunspell-en-GB	0.20121024	noarch	rpm	UK English hunspell dictionaries
alsa-plugins-pulseaudio	1.0.27	x86_64	rpm	Alsa to PulseAudio backend
xorg-x11-server-Xorg	1.17.2	x86_64	rpm	Xorg X server
emacs-filesystem	24.3	noarch	rpm	Emacs filesystem layout
openssh-clients	6.6.1p1	x86_64	rpm	An open source SSH client applications
libfreehand	0.1.1	x86_64	rpm	A library for import of Macromedia/Adobe FreeHand documents
hwloc	1.7	x86_64	rpm	Portable Hardware Locality - portable abstraction of hierarchical architecture
python-six	1.9.0	noarch	rpm	Python 2 and 3 compatibility utilities
libspectre	0.2.7	x86_64	rpm	A library for rendering PostScript(TM) documents

Windows:

-Copy the script “mk_inventory.vbs” in the “local” directory of Check_MK agent. In my case it was *C:\Program Files (x86)\check_mk\local\mk_inventory.vbs.*

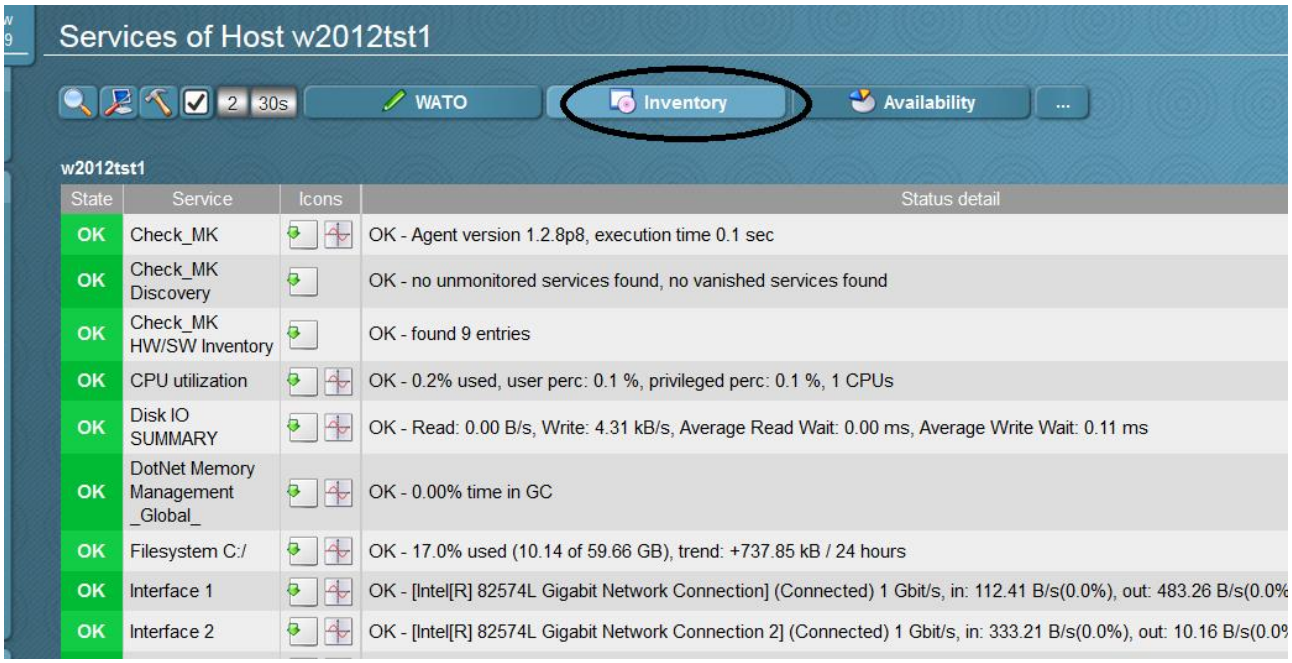
-Restart the windows service

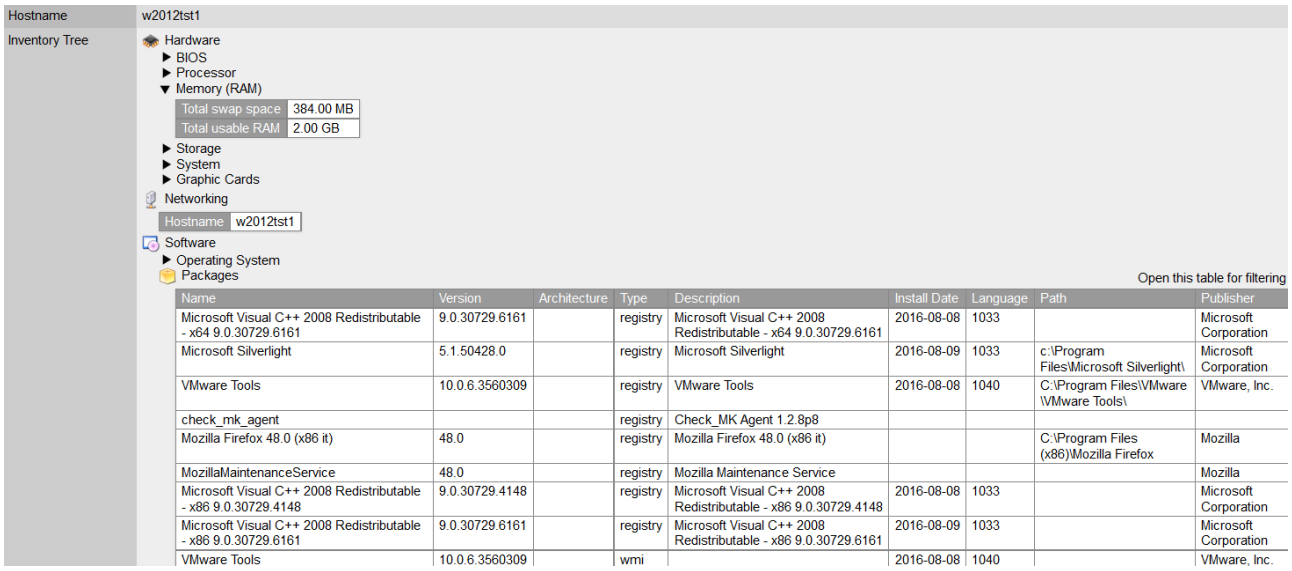
net stop check_mk_agent && net start check_mk_agent

Force inventory on the server side:

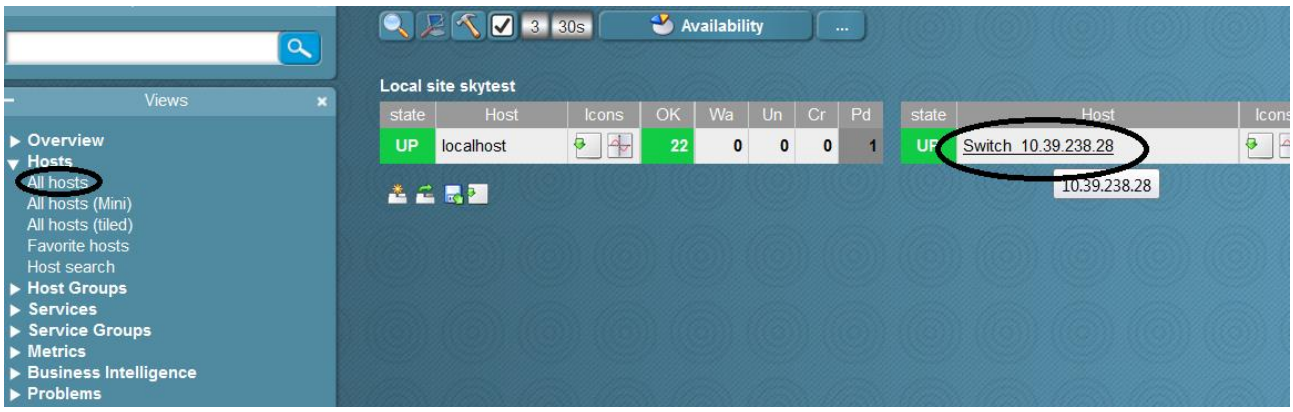
cmk –i

Click on the windows host to check what has been discovered:





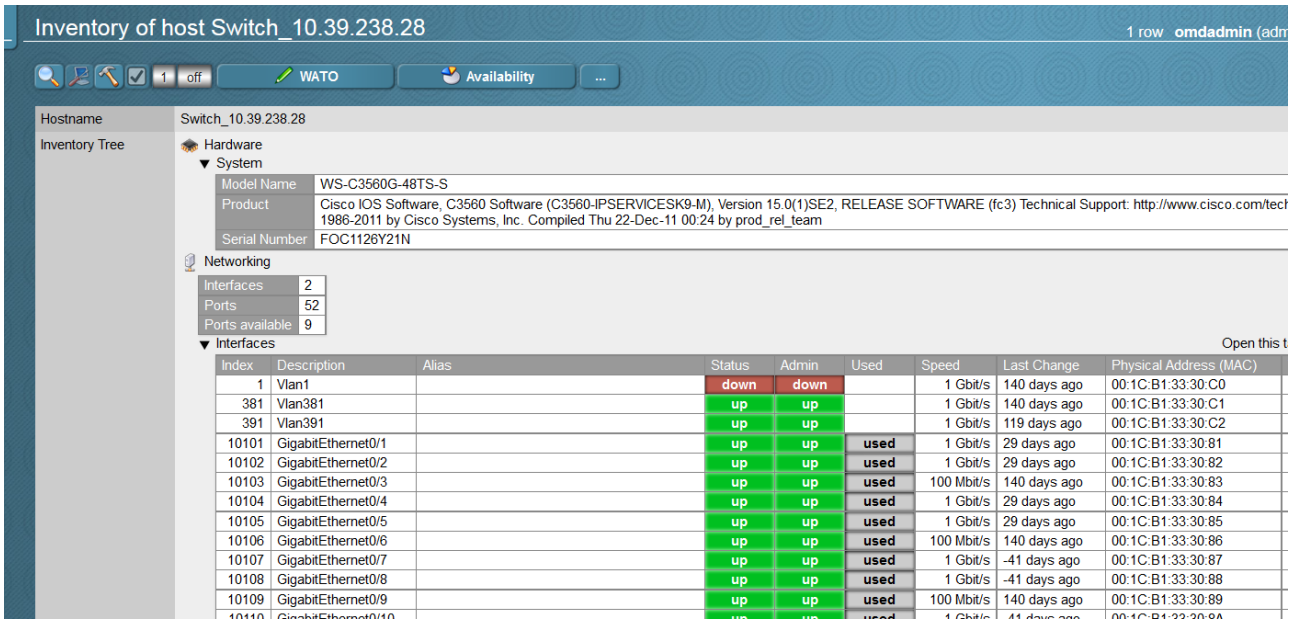
Cisco:



Click on *Inventory* button


Services of Host Switch_10.39.238.28
2
30s
WATO
Host status
Service graphs
Event History of Host
Events of Monitored ...
Inventory
Inventory History
Host downtimes
Host comments
Host Aggregations
Network Interfaces
Switch_10.39.238.28
State
Service
Icons
Status detail
OK
Check_MK
OK - execution time 1.0 sec
OK
Check_MK Discovery
OK - no unmonitored services found, no vanished services found
OK
Check_MK HW/SW Inventory
OK - found 1193 entries
OK
CPU utilization
OK - 9.0% utilization in the last 5 minutes
OK
FAN Switch 1 Fan 1
OK - State is: normal (1)


Inventory of host Switch_10.39.238.28
1
off
WATO
Availability
...
Hostname
Inventory Tree
Hardware
Networking
Software


Inventory of host Switch_10.39.238.28
1 row omdadmin
WATO
Availability
Hostname
Switch_10.39.238.28
Inventory Tree
Hardware
System
Model Name
WS-C3560G-48TS-S
Product
Cisco IOS Software, C3560 Software (C3560-IPSERVICESK9-M), Version 15.0(1)SE2, RELEASE SOFTWARE (fc3) Technical Support: http://www.cisco.com/tech
1986-2011 by Cisco Systems, Inc. Compiled Thu 22-Dec-11 00:24 by prod_rel_team
Serial Number
FOC1126Y21N
Networking
Interfaces
2
Ports
52
Ports available
9
Interfaces
Open this t
Index
Description
Alias
Status
Admin
Used
Speed
Last Change
Physical Address (MAC)
1 Vlan1 down down 1 Gbit/s 140 days ago 00:1C:B1:33:30:C0
381 Vlan381 up up 1 Gbit/s 140 days ago 00:1C:B1:33:30:C1
391 Vlan391 up up 1 Gbit/s 119 days ago 00:1C:B1:33:30:C2
10101 GigabitEthernet0/1 up up used 1 Gbit/s 29 days ago 00:1C:B1:33:30:81
10102 GigabitEthernet0/2 up up used 1 Gbit/s 29 days ago 00:1C:B1:33:30:82
10103 GigabitEthernet0/3 up up used 100 Mbit/s 140 days ago 00:1C:B1:33:30:83
10104 GigabitEthernet0/4 up up used 1 Gbit/s 29 days ago 00:1C:B1:33:30:84
10105 GigabitEthernet0/5 up up used 1 Gbit/s 29 days ago 00:1C:B1:33:30:85
10106 GigabitEthernet0/6 up up used 100 Mbit/s 140 days ago 00:1C:B1:33:30:86
10107 GigabitEthernet0/7 up up used 1 Gbit/s -41 days ago 00:1C:B1:33:30:87
10108 GigabitEthernet0/8 up up used 1 Gbit/s -41 days ago 00:1C:B1:33:30:88
10109 GigabitEthernet0/9 up up used 100 Mbit/s 140 days ago 00:1C:B1:33:30:89

Using custom plugins

Sometimes it's necessary to create custom checks and Check_MK makes this possible using *Local Checks, MRPE or MKP.*
As with *folders, Tags and Hostgroups* they are three different ways of doing the same thing and each one of them has pros and cons.

This is a summary:

Local Checks are used whenever you want something really quick and simple. Just create a script with your preferred language and place it on the monitored machine.

Pros:

- easy and asynchronous

Cons:

- no central management using WATO, all parameters will be managed inside the script.

MRPE is useful if you want a soft migration from NRPE to Check_MK.

Pros:

- supports any kind of Nagios plugin.

Cons:

- all plugins on localhost are called at the same time, once per cycle; there is no way to call some more often than others.
- The plugins are called in direct sequence - one after another. No parallelization takes place.

MKP is the native plugin format and is definitely the best/preferred method. The new packaging mechanism of Check_MK supports you in distributing your extensions and using extensions from other people by allowing you to easily create, install, update and remove packages of extensions, which are portable between all installations of Check_MK - regardless of the installations paths chosen at setup.

Pros:

- Native format, Portability, WATO support, overall efficiency

Cons:

- Requires python knowledge

Local Checks

Check_MK also has the concept of "*local checks*" that are very easy and straightforward to use and give the ability to run any kind of script or program on an agent.

Example:

a) Create a script like this and place it in the *local* directory of the Check_MK agent

```
#!/bin/bash
DIRS="/var/log /tmp"

for dir in $DIRS
do
    count=$(ls $dir | wc --lines)
    if [ $count -lt 50 ] ; then
        status=0
        statustxt=OK
    elif [ $count -lt 100 ] ; then
        status=1
        statustxt=WARNING
    else
        status=2
        statustxt=CRITICAL
    fi
    echo "$status Filecount_$dir count=$count;50;100;0; $statustxt - $count files in
$dir"
done
```

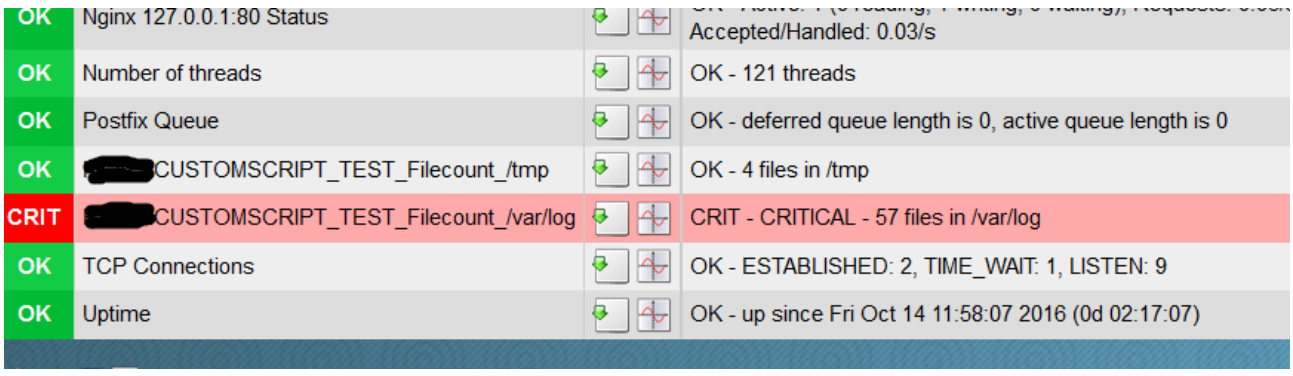
If you don't know the path to the local directory just do the following:

```
[root@centos7tst1 ~]# /usr/bin/check_mk_agent | grep -i local
Hostname: centos7tst1
LocalDirectory: /usr/lib/check_mk_agent/local
……
```

b) Do an inventory of the host running

```
cmk -I centos7tst1
```

c) The new service should show up



MRPE – Nagios Plugins

These require just a couple of steps:

a) Copy the plugin into the agent p*lugin* directory.

b) Create a configuration file *mrpe.cfg* and place it in the agent's configuration directory; if you did not change that at setup, the complete path is */etc/check_mk/mrpe.cfg.*

```
/etc/check_mk/mrpe.cfg
LOAD      /usr/lib/nagios/plugins/check_load -w 2 -c 5
FS_var    /usr/lib/nagios/plugins/check_disk /var
FS_hirn   /usr/lib/nagios/plugins/check_disk /hirn
Aptitude  /usr/lib/nagios/plugins/check_apt
Smart_sda /usr/lib/nagios/plugins/check_ide_smart -d /dev/sda -n
```

c) Inventory the host

```
cmk -I --checks=mrpe somehost123
```

MKP plugins

Instead of using *Local checks* or *MRPE,* there are lot of external plugins available in the native Check_MK format (mkp). There is a catalog on https://mathias-kettner.de/checkmk_check_catalogue.html but it's also possible to create your own using python.
To show the installation, I chose *MTR*, a nice plugin which is very useful to use when troubleshooting network problems. It was created by BenV and you can download it from his website: https://notes.benv.junerules.com/mtr/
The reason why I think this plugin is really great is that it uses *MTR*, a tool that combines the functionality of the 'traceroute' and 'ping' programs in a single network diagnostic tool.
As the documentation for *mtr* states, it investigates the network connection between the host *mtr* runs on and a user-specified destination host. After it determines the address of each network hop between the machines, it sends a sequence ICMP ECHO requests to each one to determine the quality of the link to each machine. As it does this, it prints running statistics about each machine. For more information please visit its website https://www.bitwizard.nl/mtr/

On the Check_MK host:

- Download the latest version from the website and place in /tmp

- Install using *mkp*

```
OMD[mysite]:~$ mkp install /tmp/mtr-0.5.2.mkp
```

- Copy the plugin and the configuration file onto the machine where you want to run the pings from. Please note that you need to place the plugin in the agent's *plugins* folder and the associated *cfg* file in the agent's configuration folder

```
[root@checkmktst1 tmp]# scp /opt/omd/sites/mysite/local/share/check_mk/agents/mtr
root@10.39.239.99:/usr/lib/check_mk_agent/plugins/
[root@checkmktst1 tmp]# scp
/opt/omd/sites/mysite/local/share/check_mk/agents/cfg_examples/mtr.cfg
root@10.39.239.99:/etc/check_mk/
```

On the client machine:

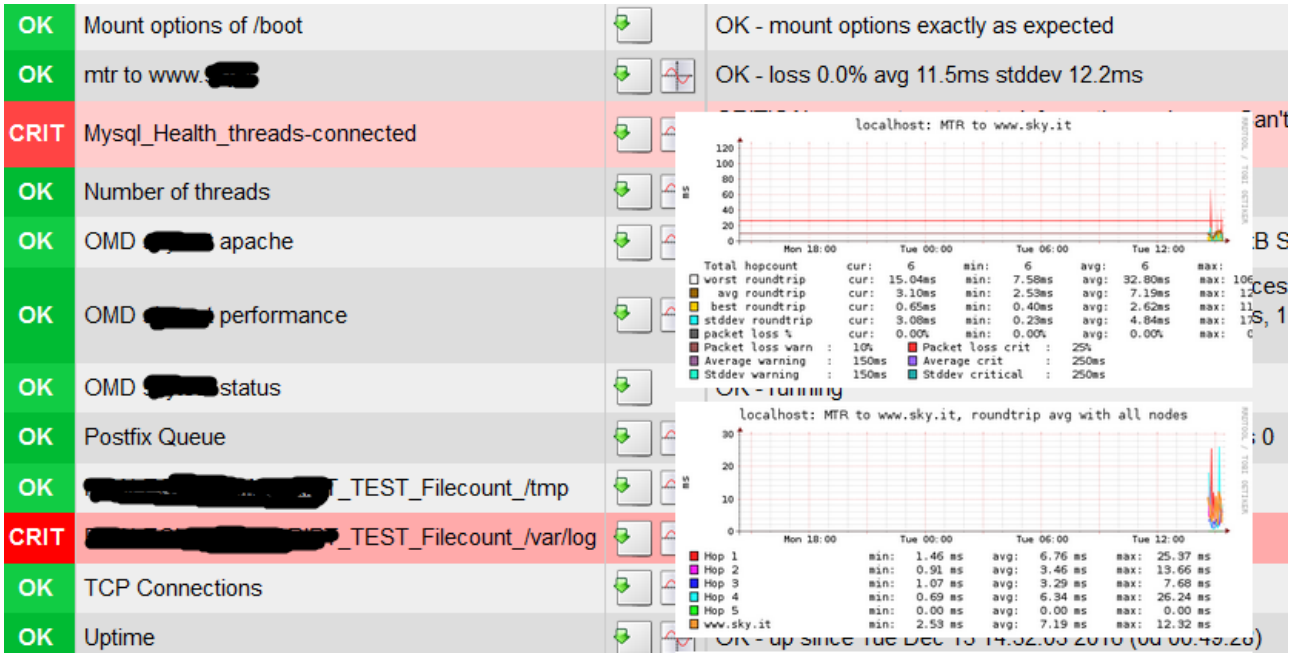
- Amend the configuration file, adding hosts that you need to monitor:

```
[root@centos7tst1 tmp]# cat /etc/check_mk/mtr.cfg
# Mtr Check_MK configuration

# NOTE: your MTR report shouldn't take longer than 15 minutes

# [DEFAULTS]
# type=icmp    # icmp, tcp or udp
# count=10     # number of pings per mtr report
# force_ipv4=0 # force ipv4, exclusive with force_ipv6
# force_ipv6=0 # force ipv6, exclusive with force_ipv4
# size=64      # packet size
# time=0         # minimum time between runs, 0 / default means run if mtr doesn't run
anymore
# port=80      # UDP/TCP port to connect to
# dns=0        # Use DNS resolution to lookup addresses
# address=     # Bind to source address
# interval=    # time MTR waits between sending pings
# timeout=     # ping Timeout, see mtr man page

[www.google.com]
type = icmp
force_ipv4 = true

[ipv6.google.com]
type = icmp
force_ipv6 = true
```

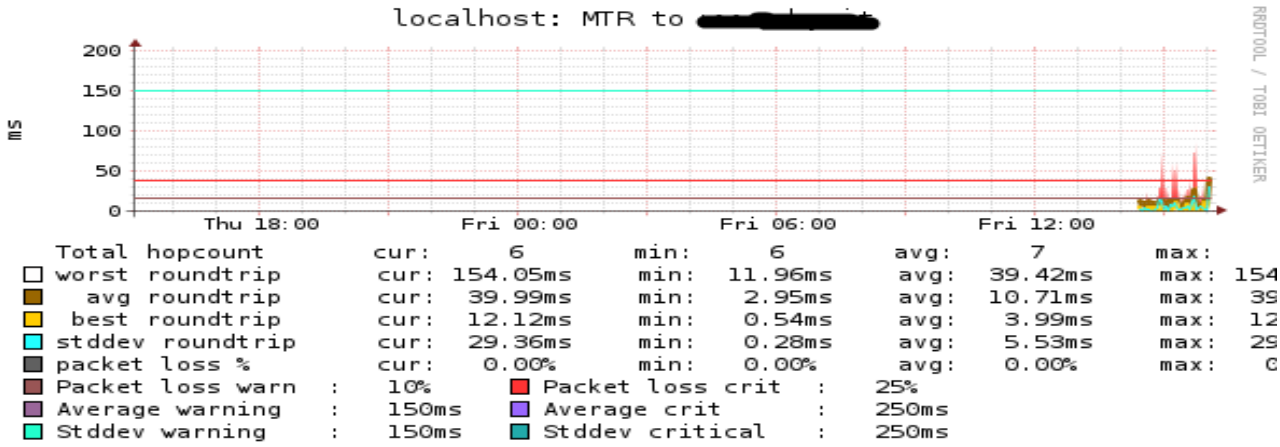
- Restart the agent

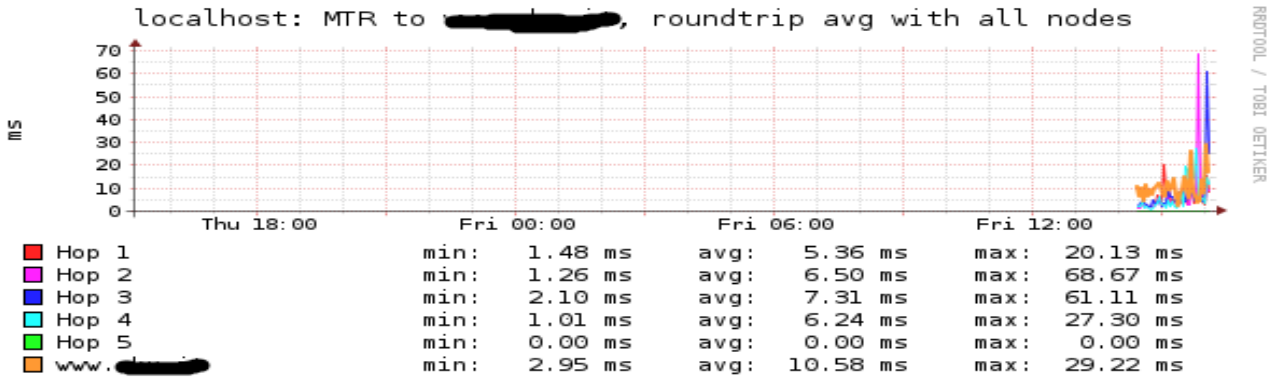
- Do a service discovery adding unmonitored services

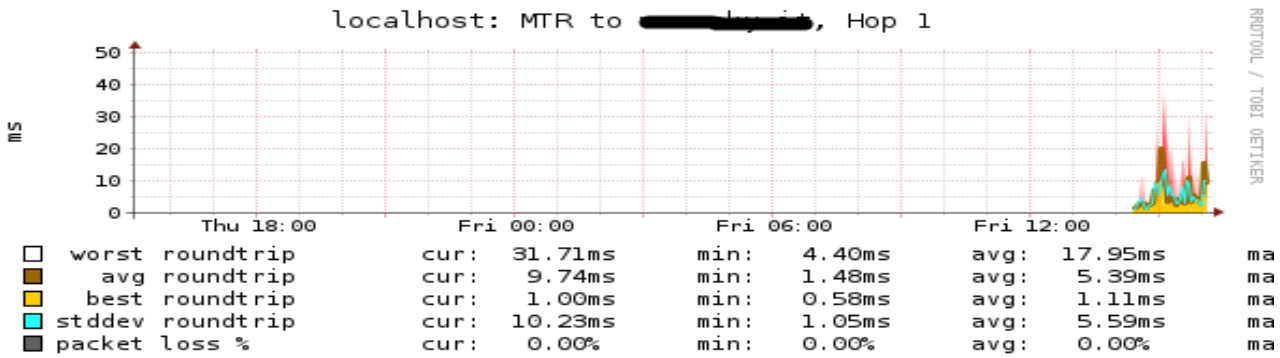
This is the result:



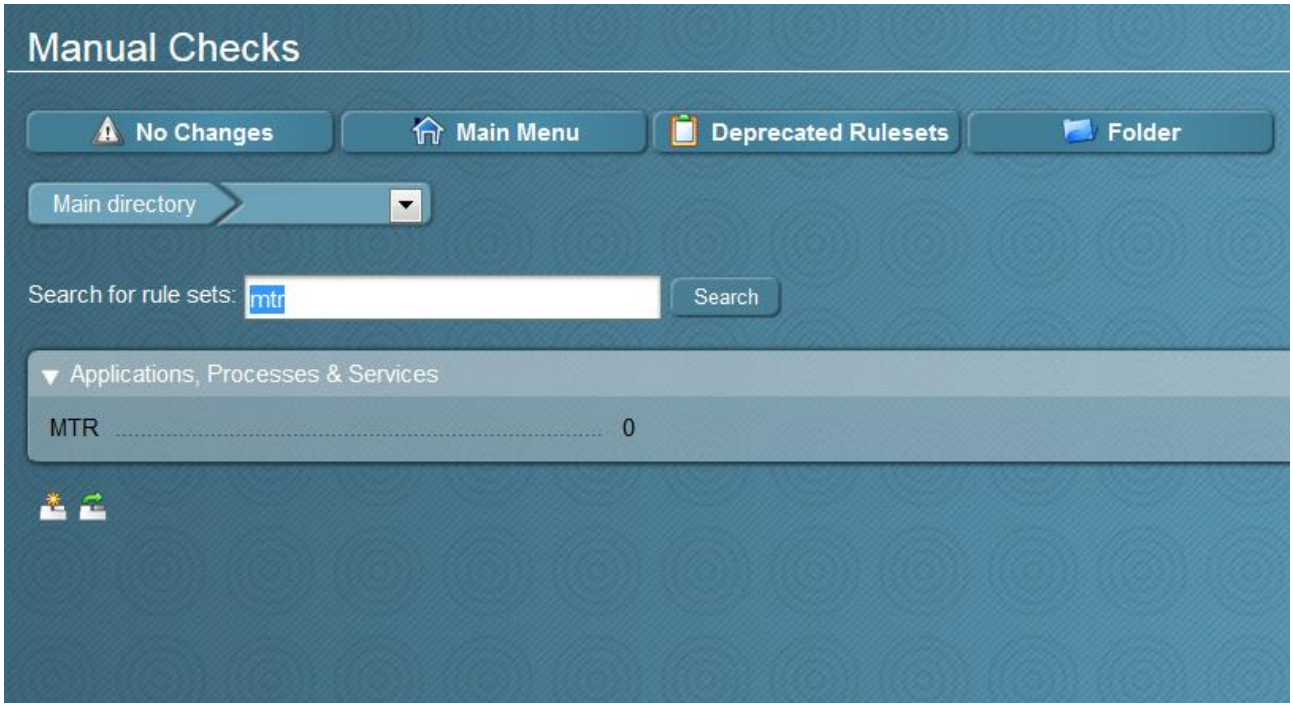
I don't think any comment is necessary here, this is really amazing!

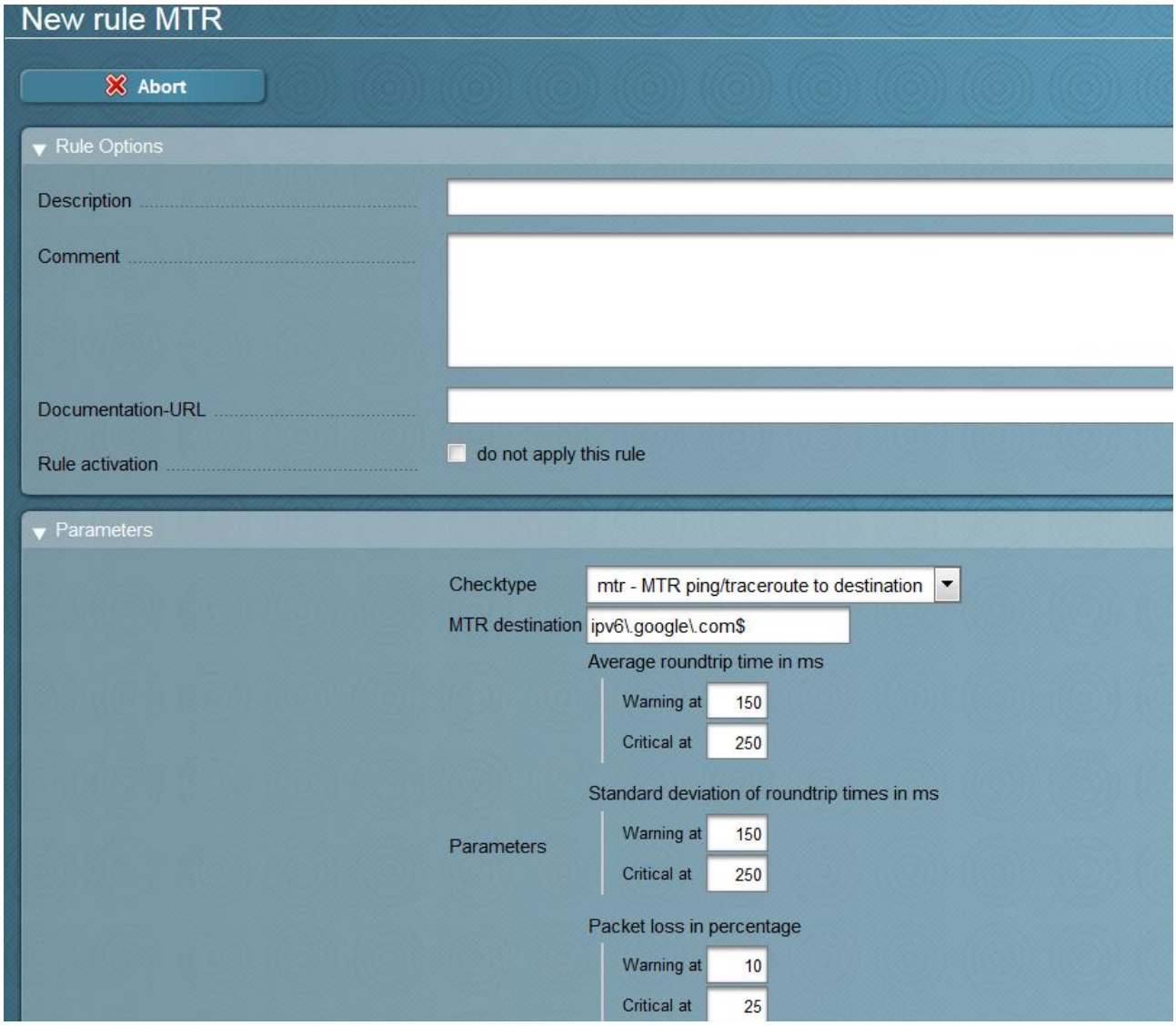






Because this is a native plugin, it's possible to manage parameters using *WATO, Manual Checks*





Monitor Apache Webserver

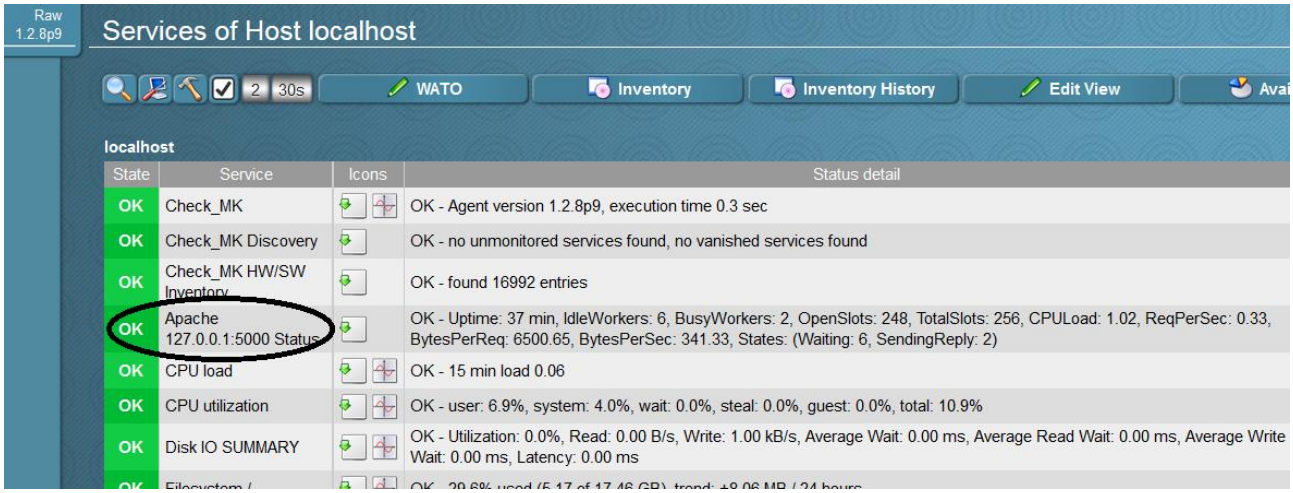
In this example, I'm going to monitor Apache using its *server-status* module that must be manually enabled in the Apache configuration file.

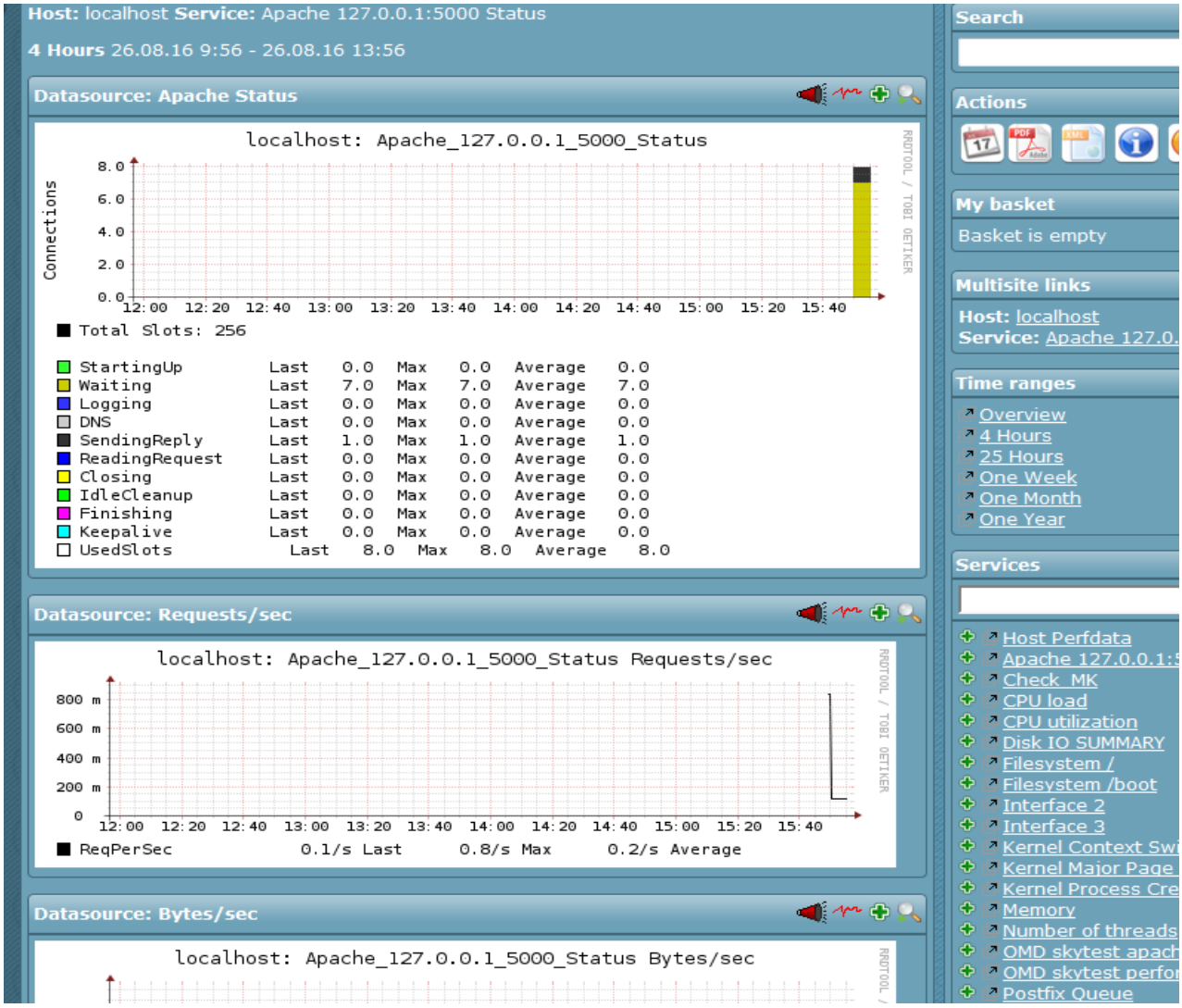
```
<IfModule mod_status.c>
    <Location /server-status>
        SetHandler server-status
        Order deny,allow
        Deny from all
        Allow from 127.0.0.1 ::1
    </Location>

    # Keep track of extended status information for each request
    ExtendedStatus On
</IfModule>
```

- Copy the apache plugin in the agent folder

```
cp -pi /opt/omd/versions/1.2.8p9.cre/share/check_mk/agents/plugins/apache_status
/usr/lib/check_mk_agent/plugins/
```

- Do a service discovery and apply changes





Monitor Mysql Server

The base agent doesn't include native support but check_mk created *mk_mysql* official plugin. I did a test on mariadb 5.5 on centos 7.2 64 bit but the same applies to other mysql versions even when running on Windows

- On mysql server, create a user only for monitoring, giving to it the rights with following SQL statement

```
MariaDB [(none)]> GRANT SELECT, SHOW DATABASES ON *.* TO 'mysqlmonitor'@'localhost'
IDENTIFIED BY 'mysqlmonitor';
```

- Copy the plugin from check_mk to the mysql host

```
[root@checkmktst1]# scp
/opt/omd/versions/1.2.8p13.cre/share/check_mk/agents/plugins/mk_mysql
root@10.39.239.99:/usr/lib/check_mk_agent/plugins
```

- Create the file *mysql.cfg* in the agent configuration folder.

```
[root@centos7tst1 ~]# cat /etc/check_mk/mysql.cfg

[client]
user=mysqlmonitor
password=mysqlmonitor
```

- Change *mysql.cfg* permissions. Setting mode 400 ensures it will not be readable for non-root users:

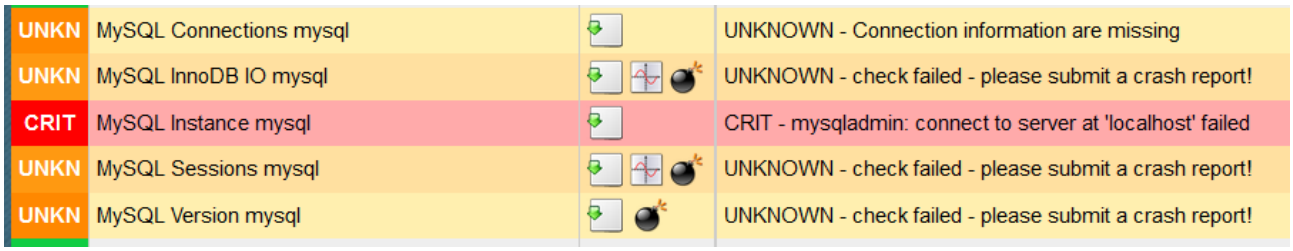
```
chmod 400 /etc/check_mk/mysql.cfg
```

- Restart the agent on the client machine

- Do a service discovery adding unmonitored services

OK	Memory		OK - RAM used: 230.75 MB of 1.80 GB, Swap used: 0.00 B of 2.00 GB, Total virtual 230.75 MB of 3.80 GB (5.9%),
OK	Mount options of /		OK - mount options exactly as expected
OK	Mount options of /boot		OK - mount options exactly as expected
OK	[redacted]		OK - loss 0.0% avg 12.3ms stddev 0.7ms
OK	MySQL Connections mysql		OK - Max. parallel Connections: 1 (Max.: 151): 0.66%
OK	MySQL InnoDB IO mysql		OK - 0.00 B/sec read, 0.00 B/sec write
OK	MySQL Instance mysql		OK - MySQL Deamon is alive
OK	MySQL Sessions mysql		OK - 1 total, 1 running, 0.44 connections/s
OK	MySQL Version mysql		OK - Version: 5.5.50-MariaDB
OK	Nginx 127.0.0.1:80 Status		OK - Active: 1 (0 reading, 1 writing, 0 waiting), Requests: 0.11/s (1.00/Connection), Accepted/Handled: 0.11/s

Whenever a Mysql fail should occur, you will be warned



As suggested in the documentation, you should also monitor the mysql process, error log, innodb redo log etc.
It's also possible to monitoring any parameter you ever want, using the community plugin available at http://exchange.check-mk.org/
To be honest, I expected something more by this plugin because Mysql is a really widespread product and there are lot of metrics that should be monitored out of the box. I'm sure that it will be improved over time but, in the meanwhile, I decided to use the excellent *check_mysql_health* from Console Labs.
There are a couple of possible paths:

- Install *check_mysql_health* on the *check_mk* host and create active checks for each parameter
- Install *check_mysql_health* directly on the mysql server and configure *MPRE*

I tested both of them but I'm going to show you only the second one because it is my preferred method
In this scenario the Mysql server's hostname is centos7tst1 with ip address 10.39.239.99

Steps:

- On the Mysql server download and Install the plugin

```
[root@centos7tst1 tmp]# wget
https://labs.consol.de/assets/downloads/nagios/check_mysql_health-2.2.2.tar.gz
[root@centos7tst1 tmp]# tar xzvf check_mysql_health-2.2.2.tar.gz
[root@centos7tst1 tmp]# cd check_mysql_health-2.2.2/
[root@centos7tst1 check_mysql_health-2.2.2]# ./configure --
prefix=/usr/lib/check_mk_agent/plugins --with-nagios-user=root --with-nagios-group=root
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
……..
……..

[root@centos7tst1 check_mysql_health-2.2.2]# make && make install
```

- Create the *MRPE* configuration file in */etc/check_mk/mrpe.cfg*

```
mysqlhealth_connection-time /usr/lib/check_mk_agent/plugins/libexec/check_mysql_health --
hostname 10.39.239.99 --username mysqlmonitor --password mysqlmonitor --mode connection-
time
mysqlhealth_uptime  /usr/lib/check_mk_agent/plugins/libexec/check_mysql_health  --hostname
10.39.239.99 --username mysqlmonitor --password mysqlmonitor --mode uptime
mysqlhealth_threads-connected  /usr/lib/check_mk_agent/plugins/libexec/check_mysql_health
--hostname 10.39.239.99 --username mysqlmonitor --password mysqlmonitor --mode threads-
connected
mysqlhealth_threadcache-hitrate
/usr/lib/check_mk_agent/plugins/libexec/check_mysql_health  --hostname  10.39.239.99  --
username mysqlmonitor --password mysqlmonitor --mode threadcache-hitrate
```

```
mysqlhealth_qcache-hitrate /usr/lib/check_mk_agent/plugins/libexec/check_mysql_health --hostname 10.39.239.99 --username mysqlmonitor --password mysqlmonitor --mode qcache-hitrate
mysqlhealth_qcache-lowmem-prunes
/usr/lib/check_mk_agent/plugins/libexec/check_mysql_health --hostname 10.39.239.99 --username mysqlmonitor --password mysqlmonitor --mode qcache-lowmem-prunes
mysqlhealth_keycache-hitrate /usr/lib/check_mk_agent/plugins/libexec/check_mysql_health --hostname 10.39.239.99 --username mysqlmonitor --password mysqlmonitor --mode keycache-hitrate
mysqlhealth_bufferpool-hitrate /usr/lib/check_mk_agent/plugins/libexec/check_mysql_health --hostname 10.39.239.99 --username mysqlmonitor --password mysqlmonitor --mode bufferpool-hitrate
mysqlhealth_bufferpool-wait-free
/usr/lib/check_mk_agent/plugins/libexec/check_mysql_health --hostname 10.39.239.99 --username mysqlmonitor --password mysqlmonitor --mode bufferpool-wait-free
mysqlhealth_log-waits    /usr/lib/check_mk_agent/plugins/libexec/check_mysql_health    --hostname 10.39.239.99 --username mysqlmonitor --password mysqlmonitor --mode log-waits
mysqlhealth_tablecache-hitrate /usr/lib/check_mk_agent/plugins/libexec/check_mysql_health --hostname 10.39.239.99 --username mysqlmonitor --password mysqlmonitor --mode tablecache-hitrate
mysqlhealth_table-lock-contention
/usr/lib/check_mk_agent/plugins/libexec/check_mysql_health --hostname 10.39.239.99 --username mysqlmonitor --password mysqlmonitor --mode table-lock-contention
mysqlhealth_index-usage   /usr/lib/check_mk_agent/plugins/libexec/check_mysql_health   --hostname 10.39.239.99 --username mysqlmonitor --password mysqlmonitor --mode index-usage
mysqlhealth_slow-queries   /usr/lib/check_mk_agent/plugins/libexec/check_mysql_health   --hostname 10.39.239.99 --username mysqlmonitor --password mysqlmonitor --mode slow-queries
mysqlhealth_long-running-procs /usr/lib/check_mk_agent/plugins/libexec/check_mysql_health --hostname 10.39.239.99 --username mysqlmonitor --password mysqlmonitor --mode long-running-procs
mysqlhealth_open-files    /usr/lib/check_mk_agent/plugins/libexec/check_mysql_health    --hostname 10.39.239.99 --username mysqlmonitor --password mysqlmonitor --mode open-files
```

- Change *Mysql* permission to allow connections coming from 10.39.239.99 that is the primary ip address of the machine.

```
MariaDB [(none)]> GRANT SELECT, SHOW DATABASES ON *.* TO 'mysqlmonitor'@'10.39.239.99' IDENTIFIED BY 'mysqlmonitor';
```

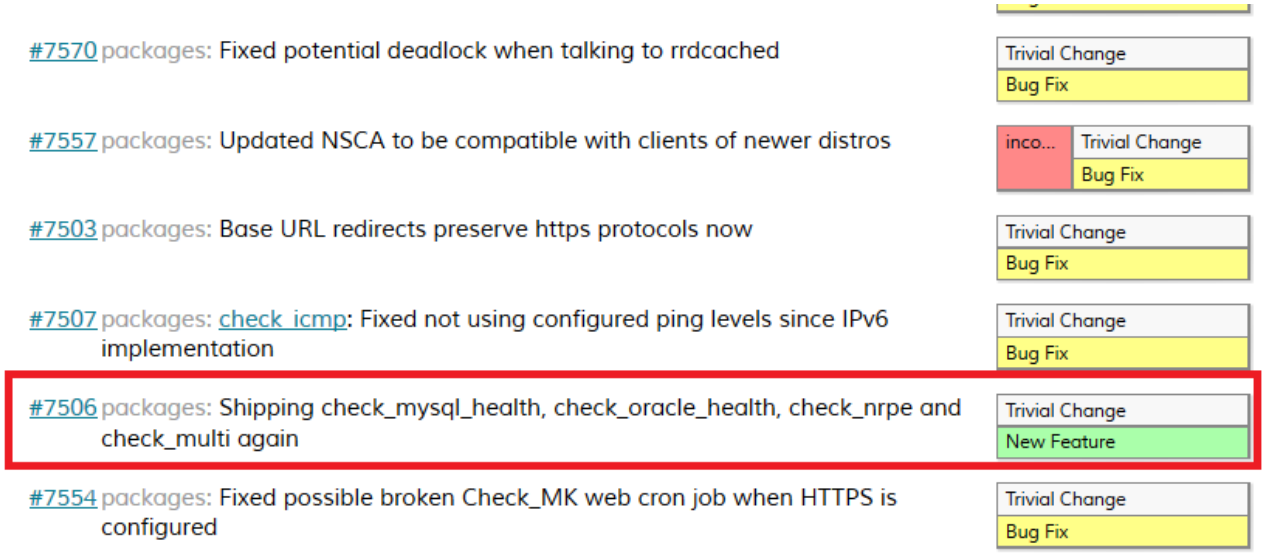
- On the *check_mk* host run a new host inventory

```
cmk –II centos7tst1
cmk -R
```

CRIT	mysqlhealth_bufferpool-hitrate		CRIT - CRITICAL - innodb buffer pool hitrate at 68.63%
OK	mysqlhealth_bufferpool-wait-free		OK - 0 innodb buffer pool waits in 60 seconds (0.0000/sec)
OK	mysqlhealth_connection-time		OK - 0.03 seconds to connect as mysqlmonitor
CRIT	mysqlhealth_index-usage		CRIT - CRITICAL - index usage 5.16%
CRIT	mysqlhealth_keycache-hitrate		CRIT - CRITICAL - myisam keycache hitrate at 50.00%
OK	mysqlhealth_log-waits		OK - 0 innodb log waits in 60 seconds (0.0000/sec)
OK	mysqlhealth_long-running-procs		OK - 0 long running processes
OK	mysqlhealth_open-files		OK - 4.98% of the open files limit reached (51 of max. 1024)
OK	mysqlhealth_qcache-hitrate		OK - query cache hitrate 0.00% (because it's turned off)
OK	mysqlhealth_qcache-lowmem-prunes		OK - 0 query cache lowmem prunes in 60 seconds (0.00/sec)
OK	mysqlhealth_slow-queries		OK - 0 slow queries in 60 seconds (0.00/sec)
OK	mysqlhealth_table-lock-contention		OK - table lock contention 0.00% (uptime < 10800)
OK	mysqlhealth_tablecache-hitrate		OK - table cache hitrate 273.33%, 10.25% filled
CRIT	mysqlhealth_threadcache-hitrate		CRIT - CRITICAL - thread cache hitrate 0.08%
OK	mysqlhealth_threads-connected		OK - 2 client connection threads
OK	mysqlhealth_uptime		OK - database is up since 65 minutes

There are lot of parameters that can be monitored and it's even possible run sql statemens using *–mode sql* defining also thresholds. Please refer to the official documentation to get more more informations.

Ps: Reading werks, I noticed that check_mysql_health should be already included in the upcoming version 1.4.



Anyway, you still need to manually install the plugin on the monitored server if you are going to use the second path

Monitor Physical Hardware

To properly monitor hardware (FAN, CPU, MEMORY, DISKs etc.) from the likes of HP or Dell, the first step is to install and configure the agents on the running OS. Because the procedure is very simple and there are many guides that show how to achieve exactly that, I'll just show the "nagios" part for an HP Proliant running Redhat 5.x

- Change *SNMPD* configuration

Because the default snmpd configuration doesn't expose all OIDs, we need to change the configuration by adding or changing the following entries:

vi /etc/snmp/snmpd.conf

```
--------snmp.conf----------

# sec.name source community
com2sec notConfigUser default public

# groupName securityModel securityName
group notConfigGroup v1 notConfigUser
group notConfigGroup v2c notConfigUser

# Make at least snmpwalk -v 1 localhost -c public system fast again.
# name incl/excl subtree mask(optional)
view all included .1
view systemview included .1.3.6.1.2.1.1
view systemview included .1.3.6.1.2.1.25.1.1

# group context sec.model sec.level prefix read write notif
access notConfigGroup "" any noauth exact all none none

---------------------------
```

- restart the snmpd service

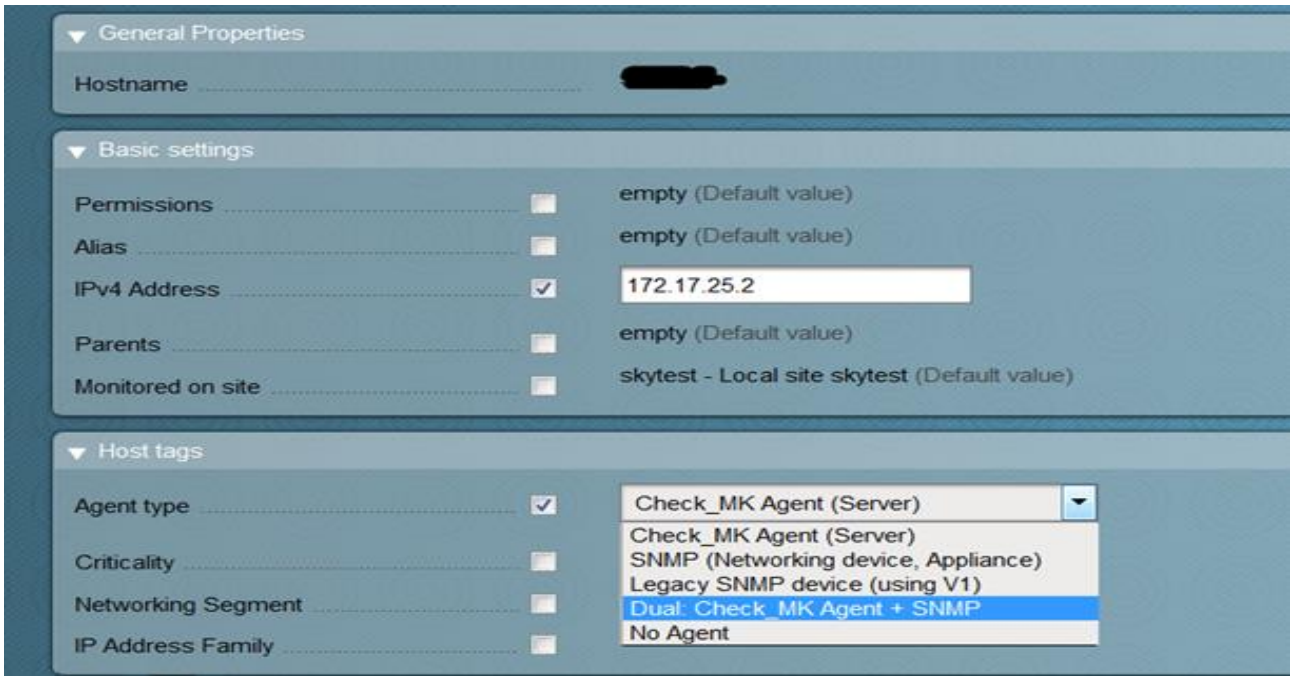
service snmpd restart

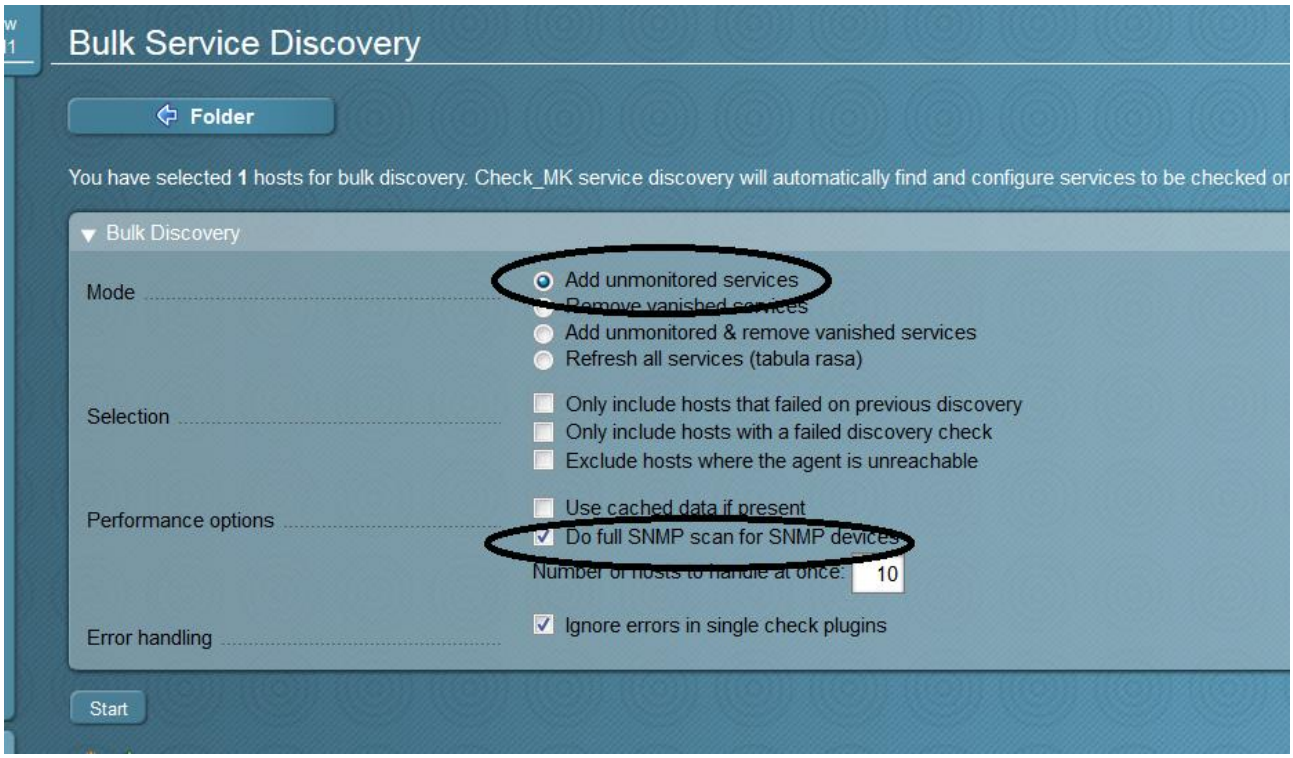
- Test the new configuration using *snmpwalk*

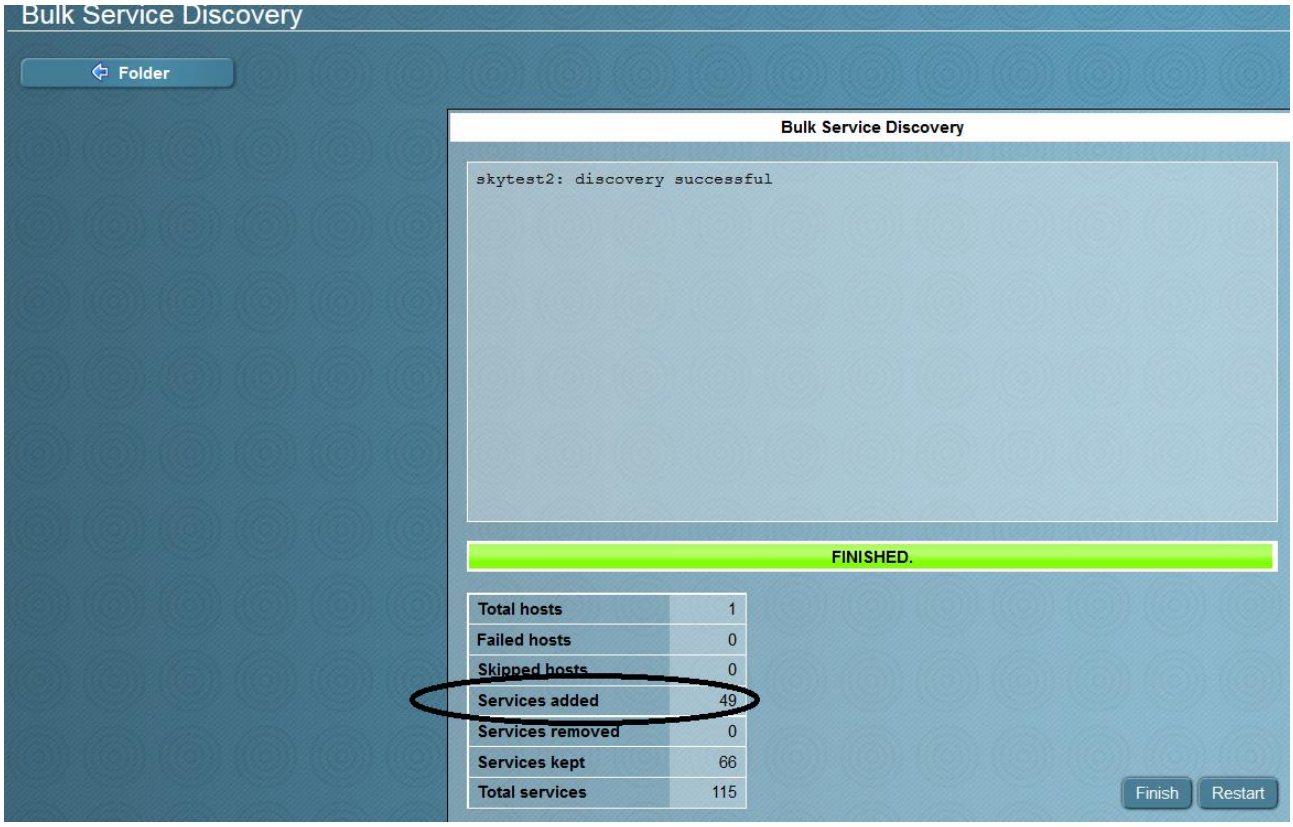
From check_MK, check if we can get the model using snmpwalk

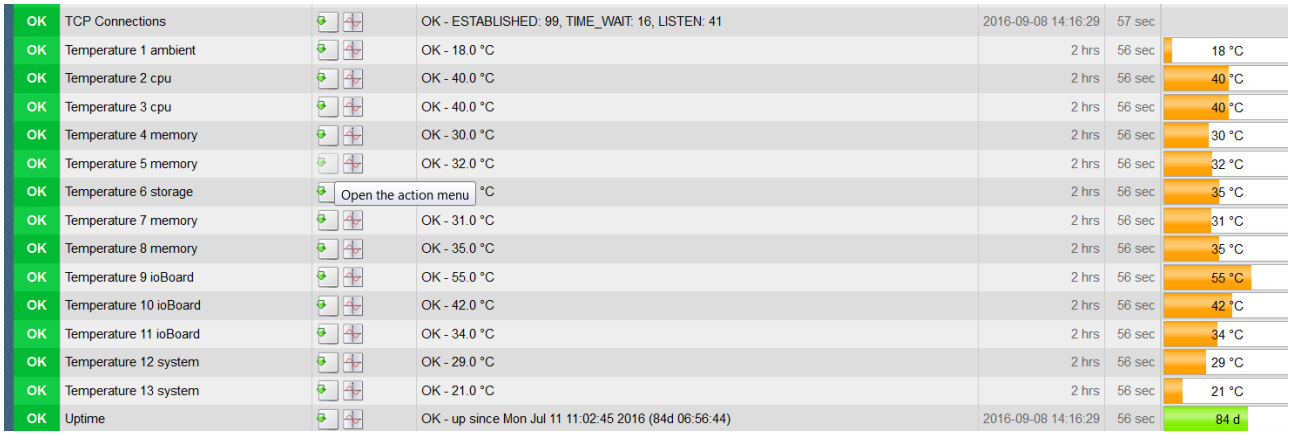
```
[root@checkmktst1 ~]# snmpwalk -v2c -c public 172.17.25.1 .1.3.6.1.4.1.232.2.2.4.2.0
SNMPv2-SMI::enterprises.232.2.2.4.2.0 = STRING: "ProLiant BL460c G7"
```

- Add the device changing the Agent type to: *Dual: Check_MK Agent + SNMP* and do a Service discovery


General Properties
Hostname
Basic settings
Permissions
empty (Default value)
Alias
empty (Default value)
IPv4 Address
172.17.25.2
Parents
empty (Default value)
Monitored on site
skytest - Local site skytest (Default value)
Host tags
Agent type
Check_MK Agent (Server)
Check_MK Agent (Server)
SNMP (Networking device, Appliance)
Legacy SNMP device (using V1)
Dual: Check_MK Agent + SNMP
No Agent
Criticality
Networking Segment
IP Address Family


Bulk Service Discovery
Folder
You have selected 1 hosts for bulk discovery. Check_MK service discovery will automatically find and configure services to be checked or
Bulk Discovery
Mode
Add unmonitored services
Add unmonitored & remove vanished services
Refresh all services (tabula rasa)
Selection
Only include hosts that failed on previous discovery
Only include hosts with a failed discovery check
Exclude hosts where the agent is unreachable
Performance options
Use cached data if present
Do full SNMP scan for SNMP devices
10
Error handling
Ignore errors in single check plugins
Start


Bulk Service Discovery
Folder
Bulk Service Discovery
skytest2: discovery successful
FINISHED.
Total hosts 1
Failed hosts 0
Skipped hosts 0
Services added 49
Services removed 0
Services kept 66
Total services 115
Finish
Restart


OK TCP Connections OK - ESTABLISHED: 99, TIME_WAIT: 16, LISTEN: 41 2016-09-08 14:16:29 57 sec
OK Temperature 1 ambient OK - 18.0 °C 2 hrs 56 sec 18 °C
OK Temperature 2 cpu OK - 40.0 °C 2 hrs 56 sec 40 °C
OK Temperature 3 cpu OK - 40.0 °C 2 hrs 56 sec 40 °C
OK Temperature 4 memory OK - 30.0 °C 2 hrs 56 sec 30 °C
OK Temperature 5 memory OK - 32.0 °C 2 hrs 56 sec 32 °C
OK Temperature 6 storage Open the action menu °C 2 hrs 56 sec 35 °C
OK Temperature 7 memory OK - 31.0 °C 2 hrs 56 sec 31 °C
OK Temperature 8 memory OK - 35.0 °C 2 hrs 56 sec 35 °C
OK Temperature 9 ioBoard OK - 55.0 °C 2 hrs 56 sec 55 °C
OK Temperature 10 ioBoard OK - 42.0 °C 2 hrs 56 sec 42 °C
OK Temperature 11 ioBoard OK - 34.0 °C 2 hrs 56 sec 34 °C
OK Temperature 12 system OK - 29.0 °C 2 hrs 56 sec 29 °C
OK Temperature 13 system OK - 21.0 °C 2 hrs 56 sec 21 °C
OK Uptime OK - up since Mon Jul 11 11:02:45 2016 (84d 06:56:44) 2016-09-08 14:16:29 56 sec 84 d

Monitor Vmware

In order to monitor VMware ESXi and vCenter Server, Check_MK has implemented a plugin that uses the vSphere API that is much more efficient that other free plugins like *check_esx3.pl* or *check_vmware_api.pl*. In the Vmware World, basically there are 2 kinds of environments:

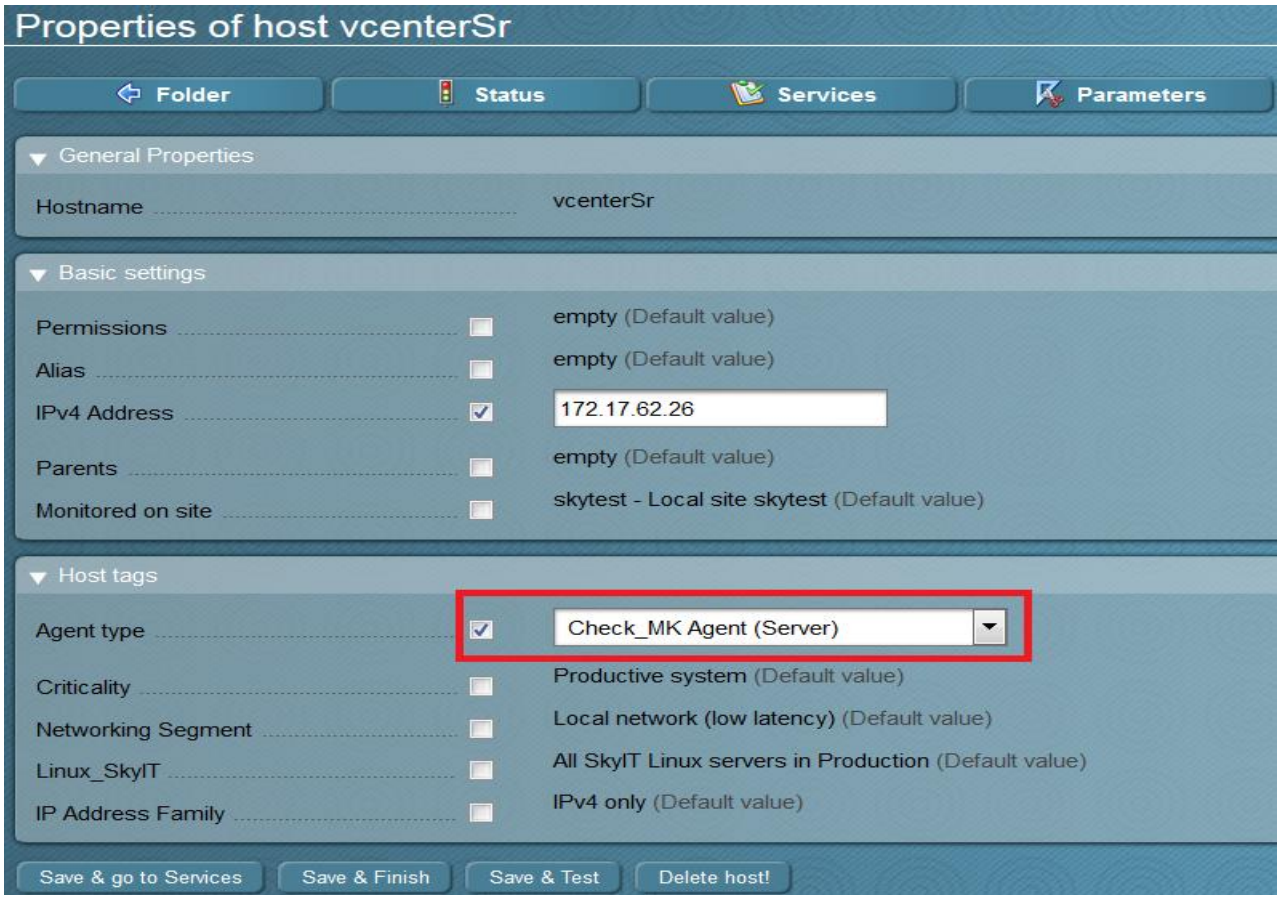
- ESXi free – Should be used just for test or lab, no support, no vcenter, no backup using external tools (apis locked out)
- vSphere that comes with different licensing options – It does include vCenter + a certain number of ESXi hosts depending on the licence

In both cases, monitoring has the following requirements:

Read-only user on vsphere side
Tcp port 443 (check_mk towards vsphere)

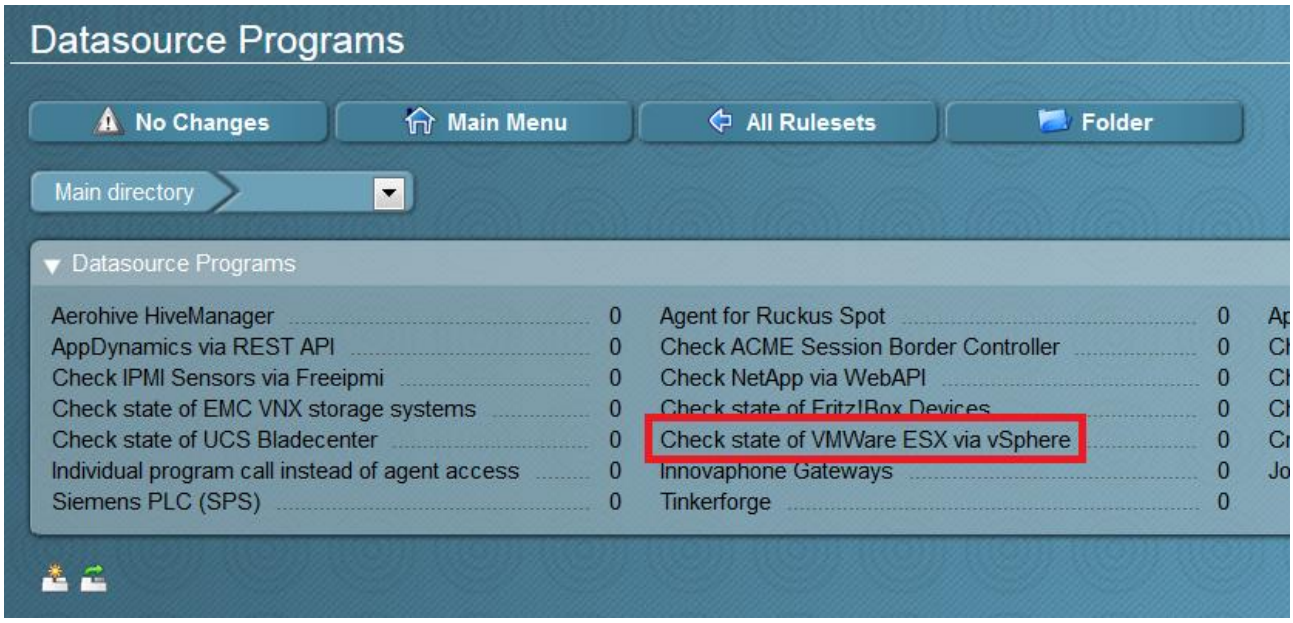
Add vSphere Virtual Center

Add the vcenter host entering the *Hostname*, *IPv4 Address* and as Agent select *Check_MK Agent* even though it isn't really installed. Click on *Save & Finish*



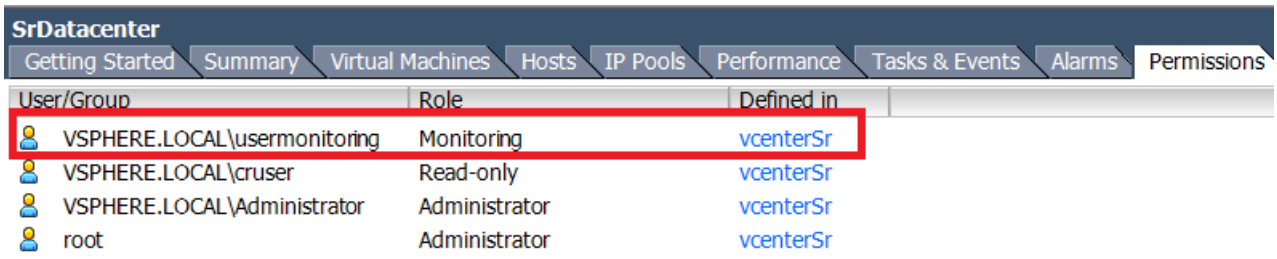
Click *Save & Finish*

To enable the advanced monitoring, in *WATO* configuration go to ***Host & Service Parameters, Datasource Programs*** and select ***Check state of VMware ESX via vSphere***.

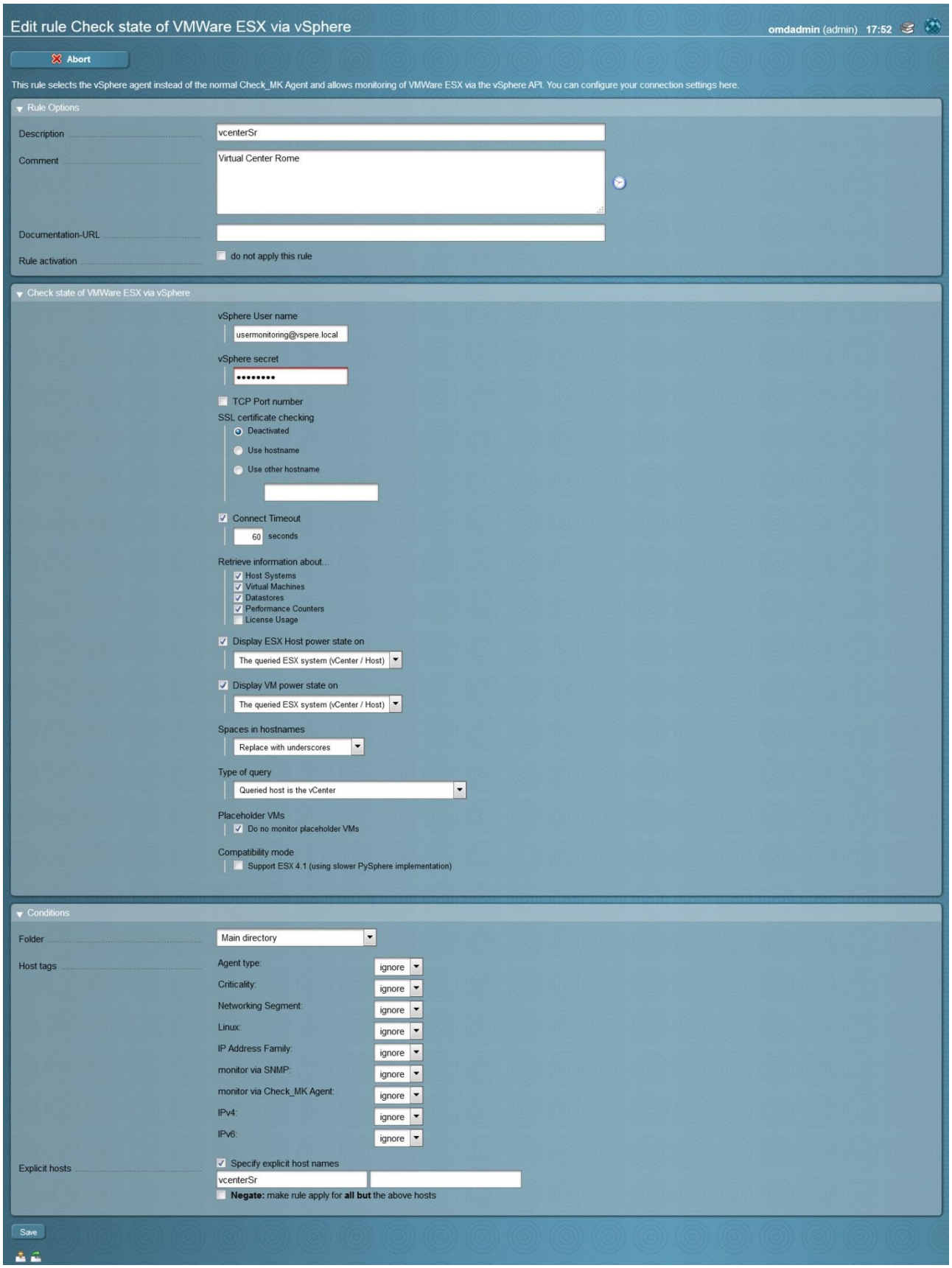


Create a new rule by clicking the button ***Create rule in folder*** and fill others fields as shown in the picture below. Just please note that:

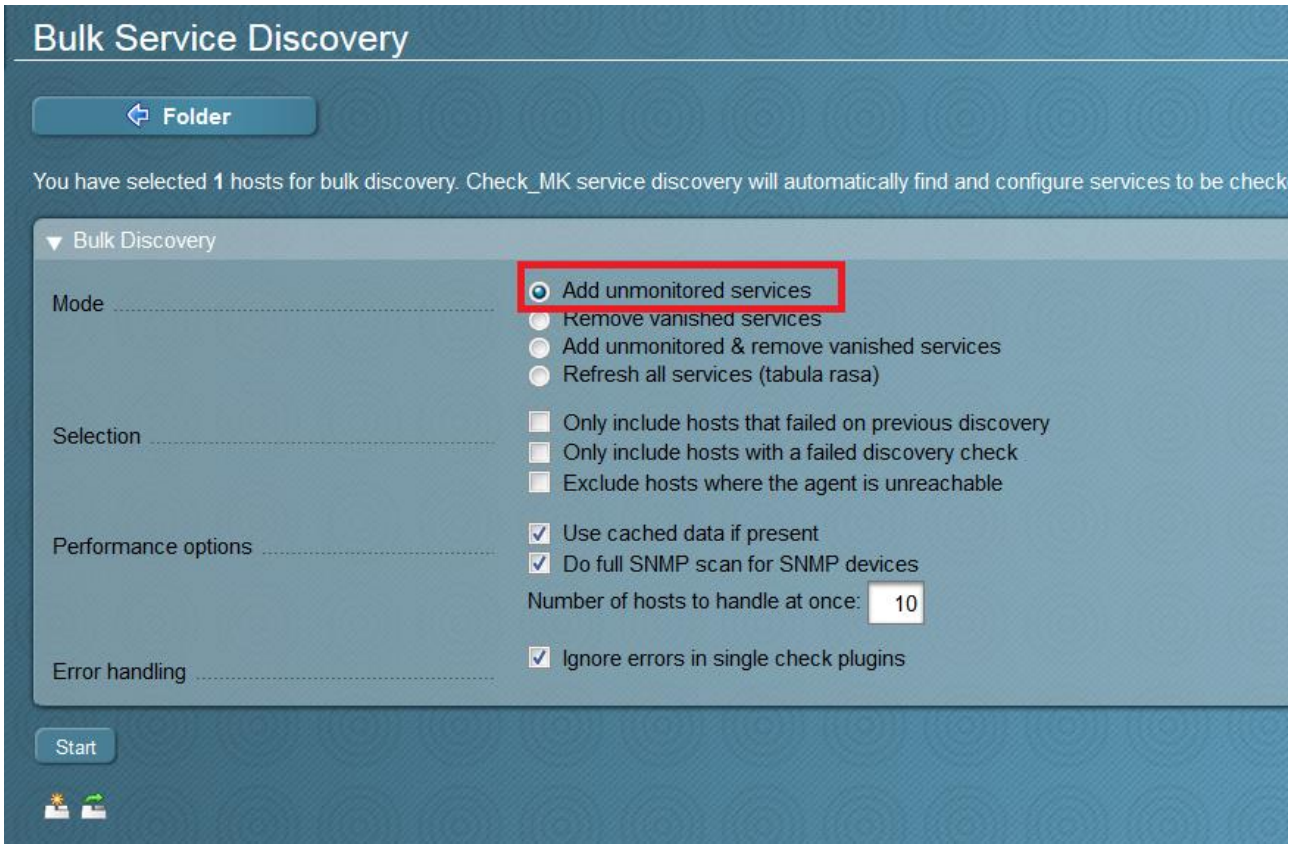
- as vSphere User Name, I created an ad-hoc user that has just read-only permission:



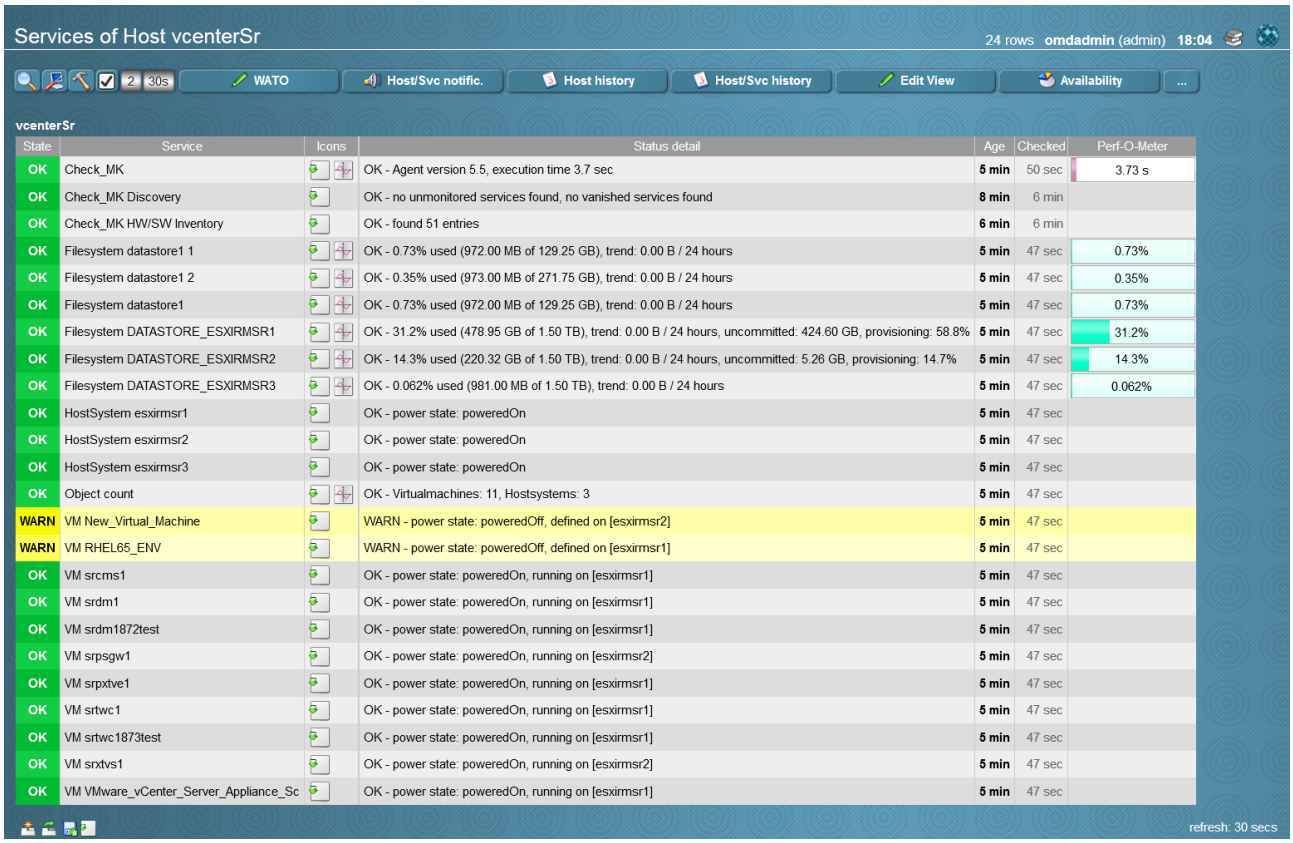
- Is possible to define which kind of informations to retrieve: *Host Systems, Virtual Machines, Datastores, Performance counters, License*.
You can select all of them at the price of a longer check execution time



Click *Save* and do a new *Bulk Service Discovery* to add unmonitored services



Activate changes and look the discovered services



State	Service	Icons	Status detail	Age	Checked	Perf-O-Meter
OK	Check_MK		OK - Agent version 5.5, execution time 3.7 sec	5 min	50 sec	3.73 s
OK	Check_MK Discovery		OK - no unmonitored services found, no vanished services found	8 min	6 min	
OK	Check_MK HW/SW Inventory		OK - found 51 entries	6 min	6 min	
OK	Filesystem datastore1 1		OK - 0.73% used (972.00 MB of 129.25 GB), trend: 0.00 B / 24 hours	5 min	47 sec	0.73%
OK	Filesystem datastore1 2		OK - 0.35% used (973.00 MB of 271.75 GB), trend: 0.00 B / 24 hours	5 min	47 sec	0.35%
OK	Filesystem datastore1		OK - 0.73% used (972.00 MB of 129.25 GB), trend: 0.00 B / 24 hours	5 min	47 sec	0.73%
OK	Filesystem DATASTORE_ESXIRMSR1		OK - 31.2% used (478.95 GB of 1.50 TB), trend: 0.00 B / 24 hours, uncommitted: 424.60 GB, provisioning: 58.8%	5 min	47 sec	31.2%
OK	Filesystem DATASTORE_ESXIRMSR2		OK - 14.3% used (220.32 GB of 1.50 TB), trend: 0.00 B / 24 hours, uncommitted: 5.26 GB, provisioning: 14.7%	5 min	47 sec	14.3%
OK	Filesystem DATASTORE_ESXIRMSR3		OK - 0.062% used (981.00 MB of 1.50 TB), trend: 0.00 B / 24 hours	5 min	47 sec	0.062%
OK	HostSystem esxirmsr1		OK - power state: poweredOn	5 min	47 sec	
OK	HostSystem esxirmsr2		OK - power state: poweredOn	5 min	47 sec	
OK	HostSystem esxirmsr3		OK - power state: poweredOn	5 min	47 sec	
OK	Object count		OK - Virtualmachines: 11, Hostsystems: 3	5 min	47 sec	
WARN	VM New_Virtual_Machine		WARN - power state: poweredOff, defined on [esxirmsr2]	5 min	47 sec	
WARN	VM RHEL65_ENV		WARN - power state: poweredOff, defined on [esxirmsr1]	5 min	47 sec	
OK	VM srcms1		OK - power state: poweredOn, running on [esxirmsr1]	5 min	47 sec	
OK	VM srdm1		OK - power state: poweredOn, running on [esxirmsr1]	5 min	47 sec	
OK	VM srdm1872test		OK - power state: poweredOn, running on [esxirmsr1]	5 min	47 sec	
OK	VM srpsgw1		OK - power state: poweredOn, running on [esxirmsr2]	5 min	47 sec	
OK	VM srpxtve1		OK - power state: poweredOn, running on [esxirmsr1]	5 min	47 sec	
OK	VM srtwc1		OK - power state: poweredOn, running on [esxirmsr1]	5 min	47 sec	
OK	VM srtwc1873test		OK - power state: poweredOn, running on [esxirmsr1]	5 min	47 sec	
OK	VM srxtvs1		OK - power state: poweredOn, running on [esxirmsr2]	5 min	47 sec	
OK	VM VMware_vCenter_Server_Appliance_Sc		OK - power state: poweredOn, running on [esxirmsr1]	5 min	47 sec	

Lot of nice informations are retrieved from vCenter such as:

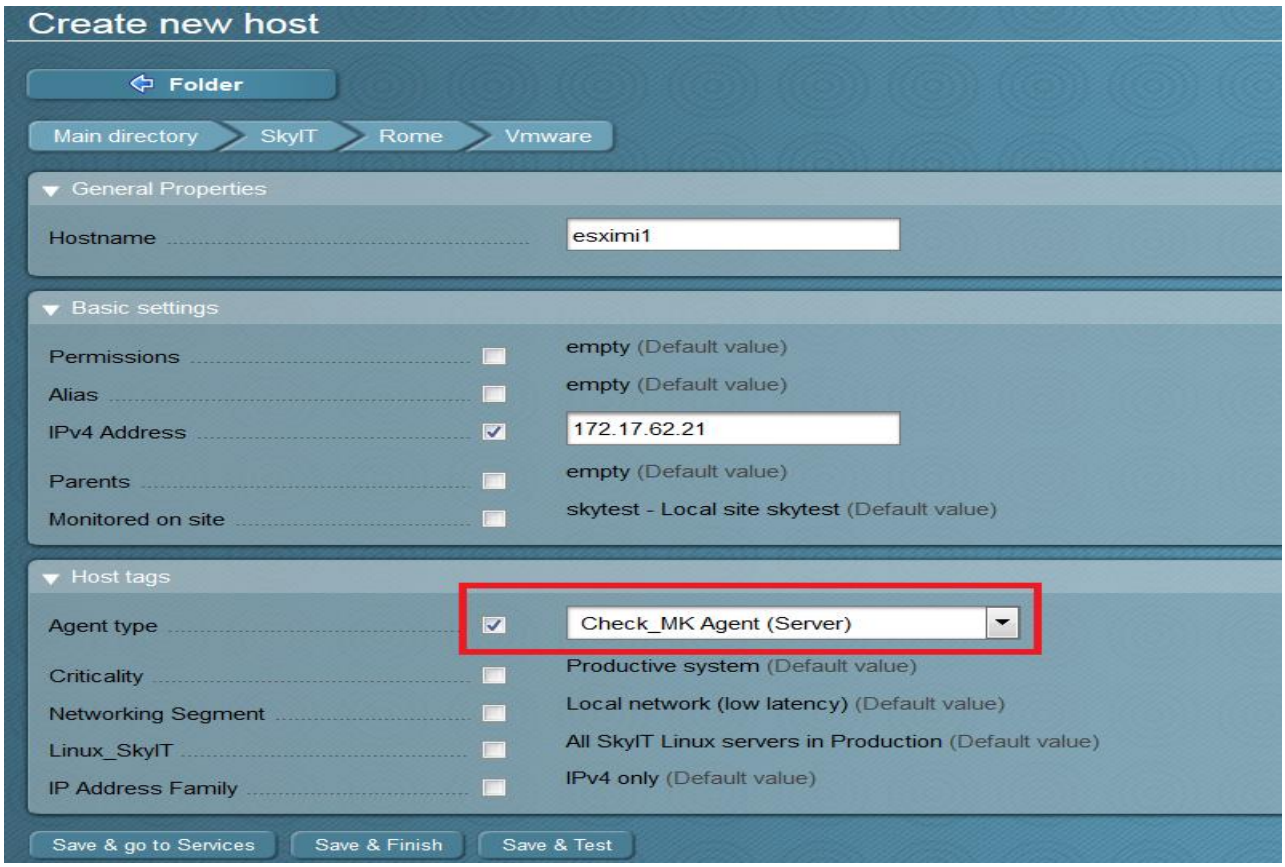
- esx_vsphere_datastores, shows all datastores (shared and local!) connected to ESXi hosts managed by the vCenter Server.
- esx_vsphere_licenses, shows all VMware licenses stored on the vCenter Server (in fact the License Manager on the Platform Services Controller)
- esx_vsphere_objects, shows connected ESXi hosts and VMs running on these hosts.

This is a basic monitoring and you could even stop here but there are a lot of precious informations missing such as interfaces usage on every single hosts, HBA status, datastore read/write/latency etc.
A good Vmware administrator should know the vital importance of these metrics, in particular the latency on datastores that caused me some headaches in the past. So let's go on adding ESXi hosts.

Note: To monitor the Vcenter host itself (like any other standard server) it is enough to install check_MK agent. Just please note that, in case of VCSA (linux virtual center appliance), we must allow incoming traffic on port 6556.
There is a step by step guide on this blog: https://paulgrevink.wordpress.com/2016/08/22/check_mk-and-vsphere-vcenter-server/

Add ESXi host managed by Vcenter

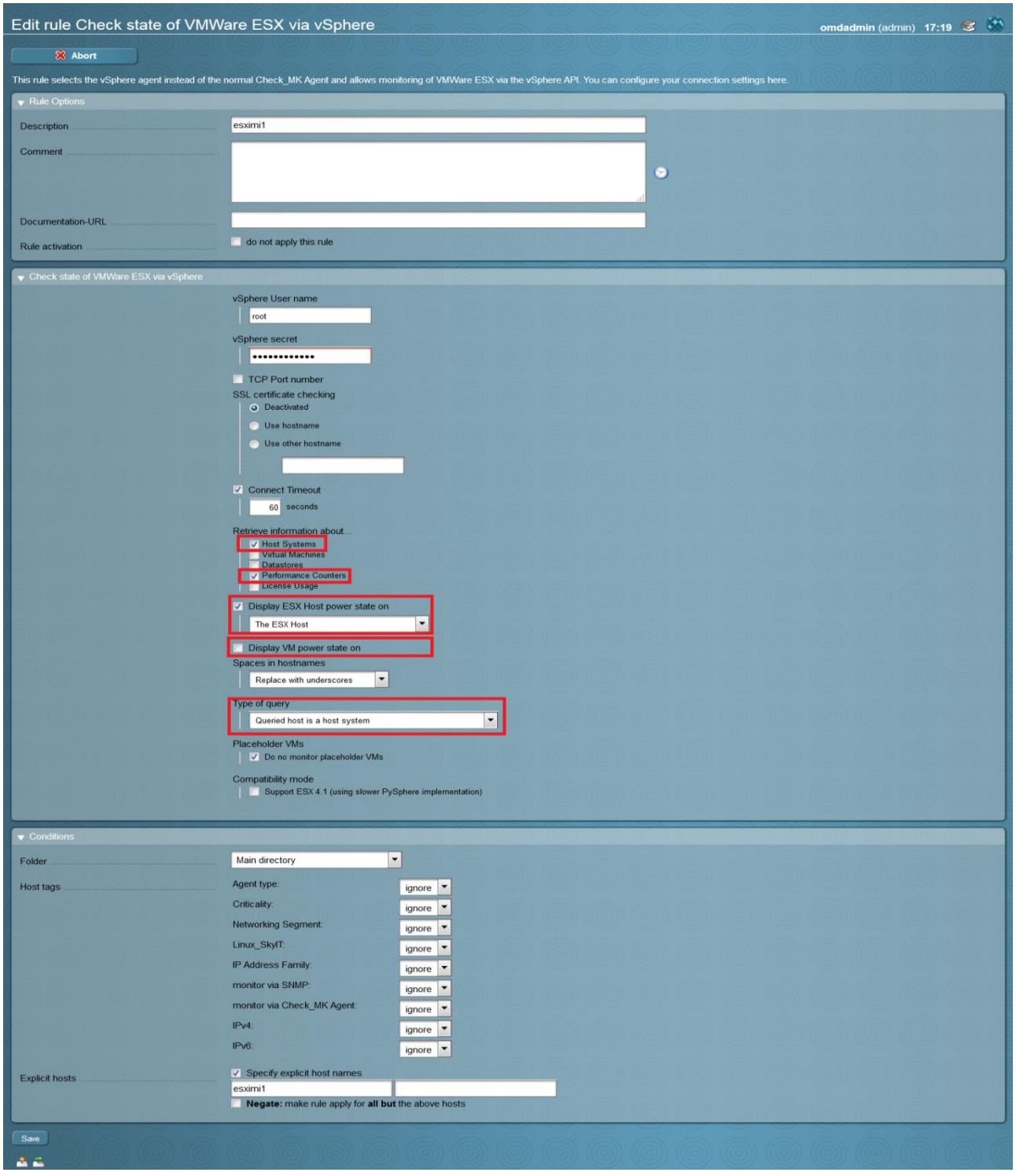
Under *WATO*, choose, *Hosts* and *New Host* enter the Hostname, IP and under ***Agent Type*** place a tick and select *Check_MK Agent*. Just please note even I'm using root, a read-only user is recommended.



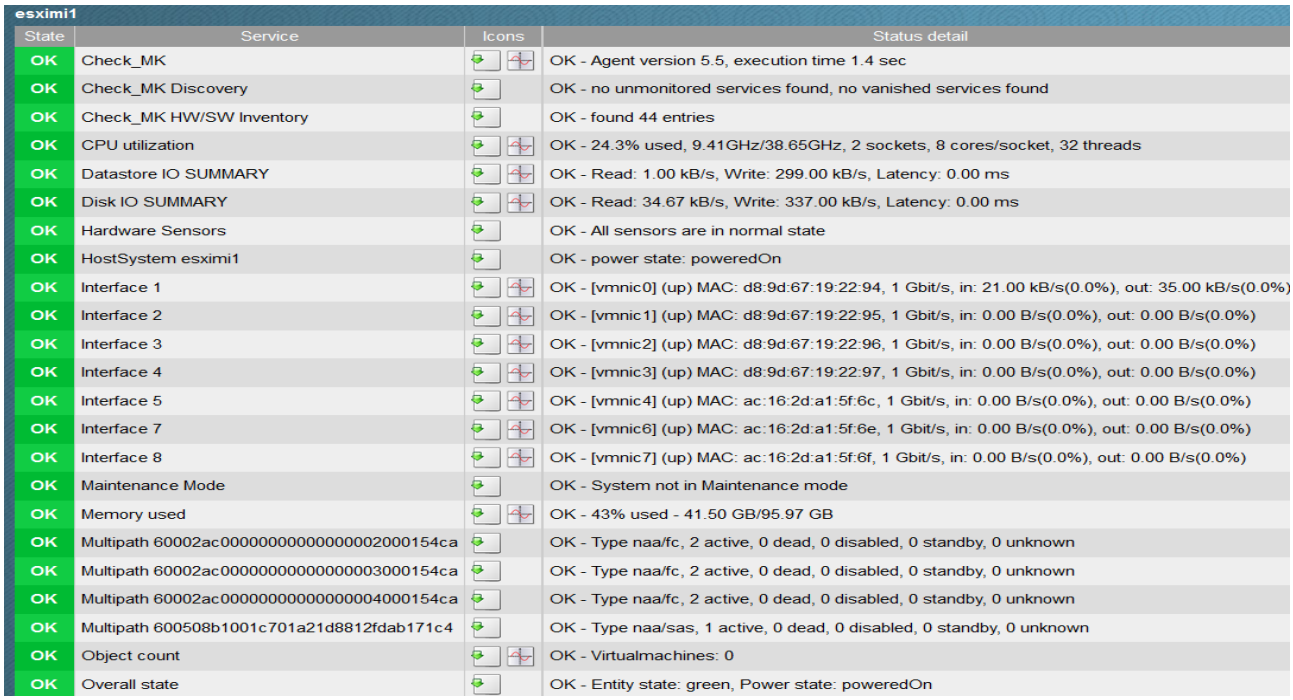
Click *Save & Finish*.

To avoid duplicated alarms, for each ESXi host managed by a vCenter Server we must create a new the rule configuring items in this way:

- *Host Systems*, Select, will show detailed status of the ESXi host.
- *Virtual Machines*, do not Select, already set on the vCenter Server.
- *Datastores*, do not Select, already set on the vCenter Server.
- *Performance Counters*, Select, will show performance counters of the ESXi hosts.
- *License Usage*, do not Select.



Do a service discovery adding unmonitored services and activate changes. Host's specific informations such as Cpu/Memory, Datastore read/write/latency and network interfaces and HBA status will be displayed.

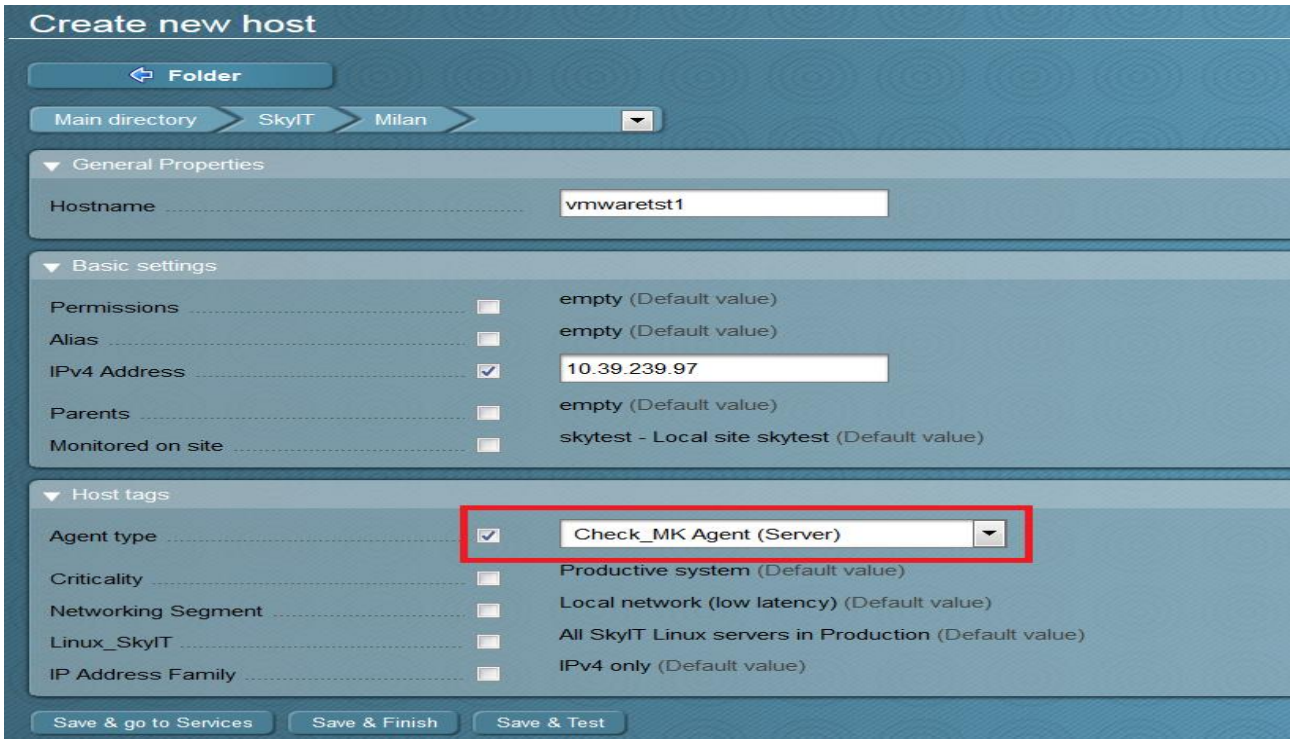


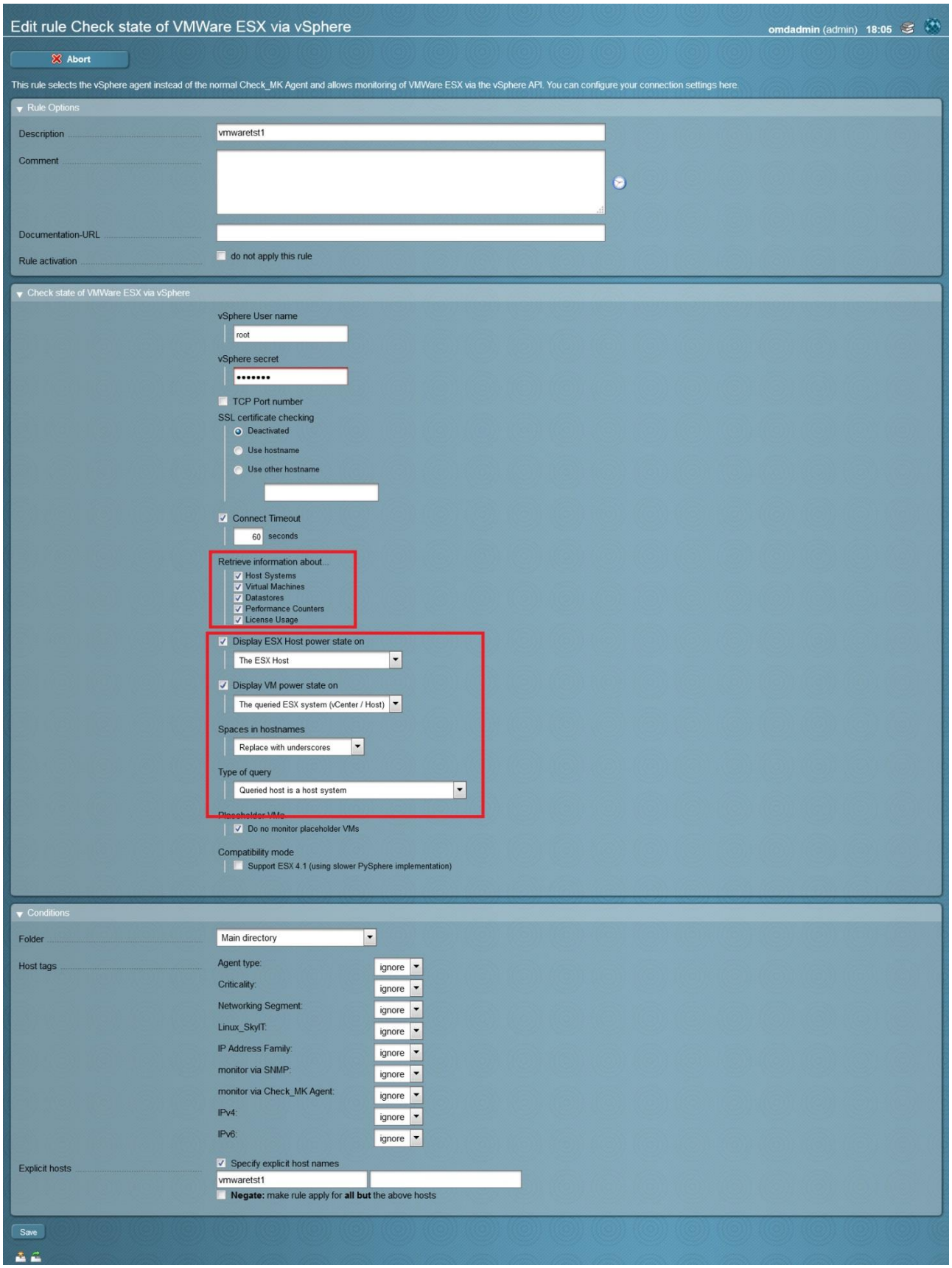
esximi1

State	Service	Icons	Status detail
OK	Check_MK		OK - Agent version 5.5, execution time 1.4 sec
OK	Check_MK Discovery		OK - no unmonitored services found, no vanished services found
OK	Check_MK HW/SW Inventory		OK - found 44 entries
OK	CPU utilization		OK - 24.3% used, 9.41GHz/38.65GHz, 2 sockets, 8 cores/socket, 32 threads
OK	Datastore IO SUMMARY		OK - Read: 1.00 kB/s, Write: 299.00 kB/s, Latency: 0.00 ms
OK	Disk IO SUMMARY		OK - Read: 34.67 kB/s, Write: 337.00 kB/s, Latency: 0.00 ms
OK	Hardware Sensors		OK - All sensors are in normal state
OK	HostSystem esximi1		OK - power state: poweredOn
OK	Interface 1		OK - [vmnic0] (up) MAC: d8:9d:67:19:22:94, 1 Gbit/s, in: 21.00 kB/s(0.0%), out: 35.00 kB/s(0.0%)
OK	Interface 2		OK - [vmnic1] (up) MAC: d8:9d:67:19:22:95, 1 Gbit/s, in: 0.00 B/s(0.0%), out: 0.00 B/s(0.0%)
OK	Interface 3		OK - [vmnic2] (up) MAC: d8:9d:67:19:22:96, 1 Gbit/s, in: 0.00 B/s(0.0%), out: 0.00 B/s(0.0%)
OK	Interface 4		OK - [vmnic3] (up) MAC: d8:9d:67:19:22:97, 1 Gbit/s, in: 0.00 B/s(0.0%), out: 0.00 B/s(0.0%)
OK	Interface 5		OK - [vmnic4] (up) MAC: ac:16:2d:a1:5f:6c, 1 Gbit/s, in: 0.00 B/s(0.0%), out: 0.00 B/s(0.0%)
OK	Interface 7		OK - [vmnic6] (up) MAC: ac:16:2d:a1:5f:6e, 1 Gbit/s, in: 0.00 B/s(0.0%), out: 0.00 B/s(0.0%)
OK	Interface 8		OK - [vmnic7] (up) MAC: ac:16:2d:a1:5f:6f, 1 Gbit/s, in: 0.00 B/s(0.0%), out: 0.00 B/s(0.0%)
OK	Maintenance Mode		OK - System not in Maintenance mode
OK	Memory used		OK - 43% used - 41.50 GB/95.97 GB
OK	Multipath 60002ac0000000000000002000154ca		OK - Type naa/fc, 2 active, 0 dead, 0 disabled, 0 standby, 0 unknown
OK	Multipath 60002ac0000000000000003000154ca		OK - Type naa/fc, 2 active, 0 dead, 0 disabled, 0 standby, 0 unknown
OK	Multipath 60002ac0000000000000004000154ca		OK - Type naa/fc, 2 active, 0 dead, 0 disabled, 0 standby, 0 unknown
OK	Multipath 600508b1001c701a21d8812fdab171c4		OK - Type naa/sas, 1 active, 0 dead, 0 disabled, 0 standby, 0 unknown
OK	Object count		OK - Virtualmachines: 0
OK	Overall state		OK - Entity state: green, Power state: poweredOn

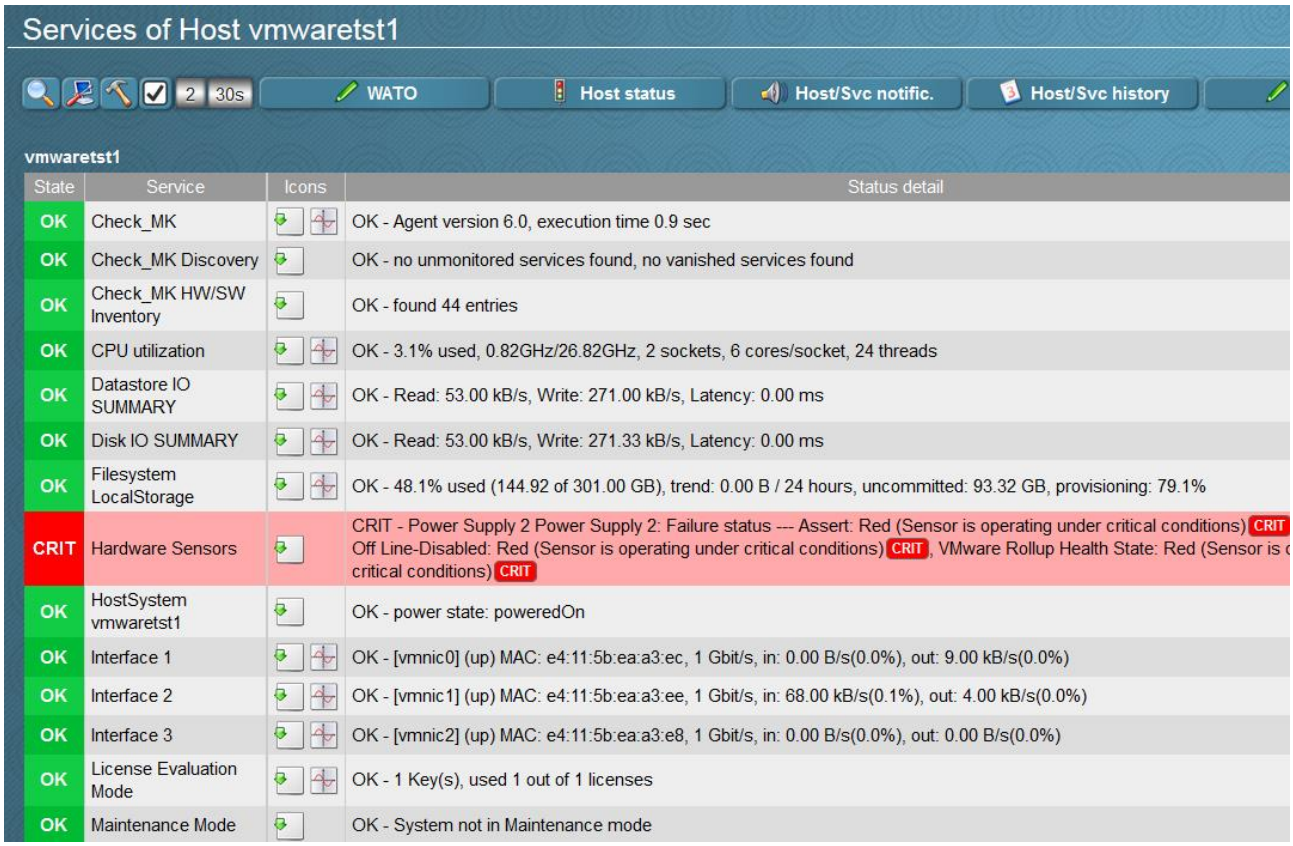
Add standalone ESXi hosts

The procedure is pretty much the same as that used to add hosts managed by the vCenter apart that all options have to be selected during the ruleset creation .




Edit rule Check state of VMWare ESX via vSphere
omdadmin (admin) 18:05
Abort
This rule selects the vSphere agent instead of the normal Check_MK Agent and allows monitoring of VMWare ESX via the vSphere API. You can configure your connection settings here.
Rule Options
Description
vmwaretst1
Comment
Documentation-URL
Rule activation
do not apply this rule
Check state of VMWare ESX via vSphere
vSphere User name
root
vSphere secret
TCP Port number
SSL certificate checking
Deactivated
Use hostname
Use other hostname
Connect Timeout
60 seconds
Retrieve information about...
Host Systems
Virtual Machines
Datastores
Performance Counters
License Usage
Display ESX Host power state on
The ESX Host
Display VM power state on
The queried ESX system (vCenter / Host)
Spaces in hostnames
Replace with underscores
Type of query
Queried host is a host system
Placeholder VMs
Do no monitor placeholder VMs
Compatibility mode
Support ESX 4.1 (using slower PySphere implementation)
Conditions
Folder
Main directory
Host tags
Agent type:
Criticality:
Networking Segment:
Linux_SkyIT:
IP Address Family:
monitor via SNMP:
monitor via Check_MK Agent:
IPv4:
IPv6:
ignore
Explicit hosts
Specify explicit host names
vmwaretst1
Negate: make rule apply for all but the above hosts
Save

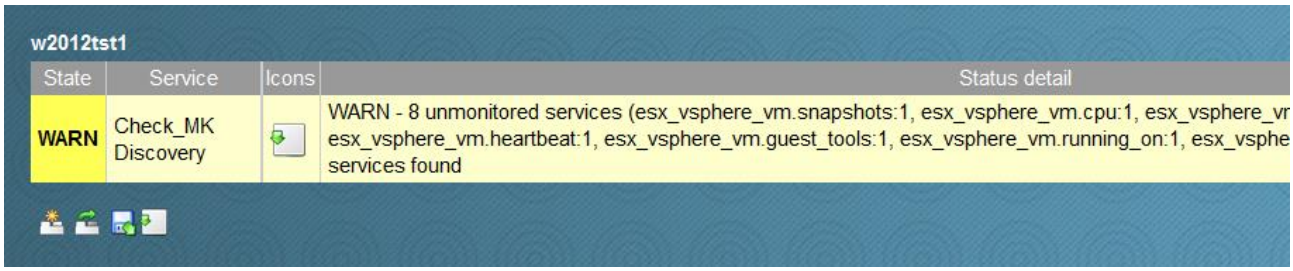
This is a standalone hp dl 360g7 running ESXi free. The critical service is related to a power supply in failed state.



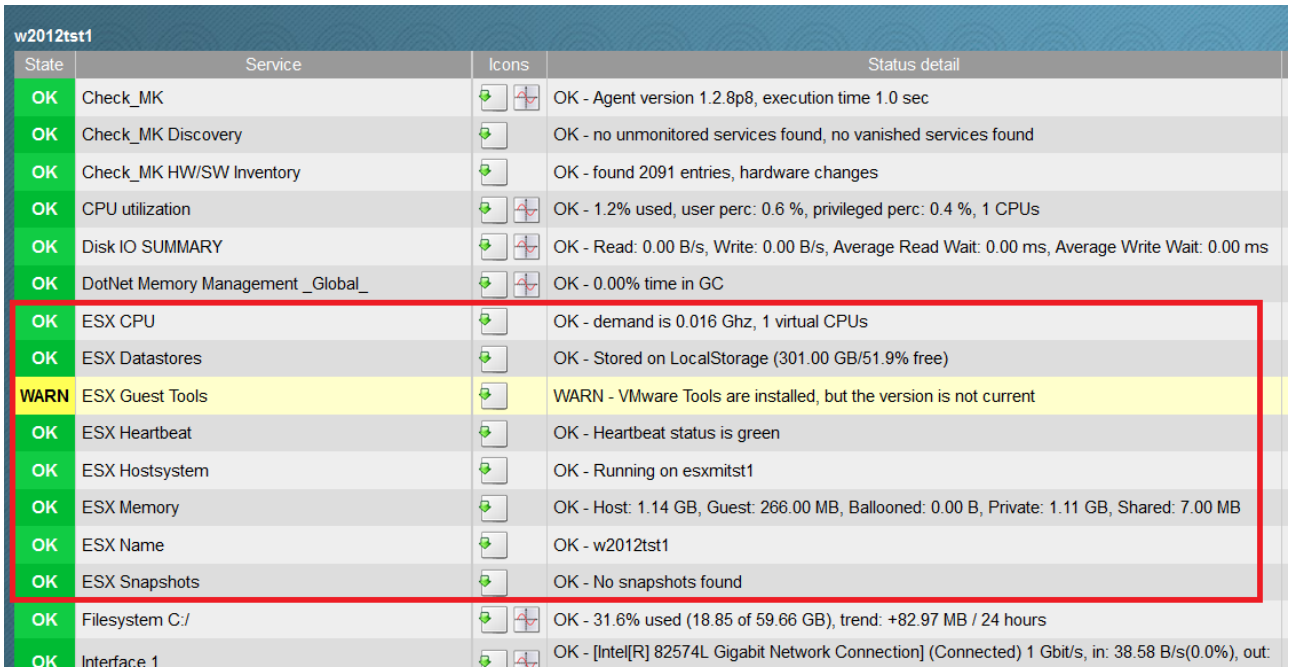
Virtual Machines additional checks

As soon as you will install check_mk agent on virtual machines, additional checks we'll added and, a great thing about that, is that performance metrics (cpu/ram) will be retrieved directly from vcenter or ESXi host and not from the OS. This is very important because in a Vmware environment, whenever you look at performance, what it really important is to know the real resources assigned by the host and not those that OS believe to have. A good example is cpu ready where the guest report high cpu usage but in reality it isn't having the right resources because there is competition on the host side. I won't go through the details because this is out of topic; if you want more informations about that, please have a look at the following link: http://www.logicmonitor.com/blog/2013/02/25/a-tale-of-two-metrics-windows-cpu-or-vcenter-vm-cpu/

After the agent installation on the guest, check_MK warned me about some missing services



Doing a new service discovery, they immediately appeared



w2012tst1

State	Service	Icons	Status detail
OK	Check_MK		OK - Agent version 1.2.8p8, execution time 1.0 sec
OK	Check_MK Discovery		OK - no unmonitored services found, no vanished services found
OK	Check_MK HW/SW Inventory		OK - found 2091 entries, hardware changes
OK	CPU utilization		OK - 1.2% used, user perc: 0.6 %, privileged perc: 0.4 %, 1 CPUs
OK	Disk IO SUMMARY		OK - Read: 0.00 B/s, Write: 0.00 B/s, Average Read Wait: 0.00 ms, Average Write Wait: 0.00 ms
OK	DotNet Memory Management _Global_		OK - 0.00% time in GC
OK	ESX CPU		OK - demand is 0.016 Ghz, 1 virtual CPUs
OK	ESX Datastores		OK - Stored on LocalStorage (301.00 GB/51.9% free)
WARN	ESX Guest Tools		WARN - VMware Tools are installed, but the version is not current
OK	ESX Heartbeat		OK - Heartbeat status is green
OK	ESX Hostsystem		OK - Running on esxmitst1
OK	ESX Memory		OK - Host: 1.14 GB, Guest: 266.00 MB, Ballooned: 0.00 B, Private: 1.11 GB, Shared: 7.00 MB
OK	ESX Name		OK - w2012tst1
OK	ESX Snapshots		OK - No snapshots found
OK	Filesystem C:/		OK - 31.6% used (18.85 of 59.66 GB), trend: +82.97 MB / 24 hours
OK	Interface 1		OK - [Intel[R] 82574L Gigabit Network Connection] (Connected) 1 Gbit/s, in: 38.58 B/s(0.0%), out:

Managing SNMP Traps

Nowadays, every good Enterprise monitoring solution has the ability to manage incoming SNMP Traps but some do it better than others. I had a frustrating experience with some tools but Check_MK, as usual, does it really well and in a clear and simple way.

Our goal is:

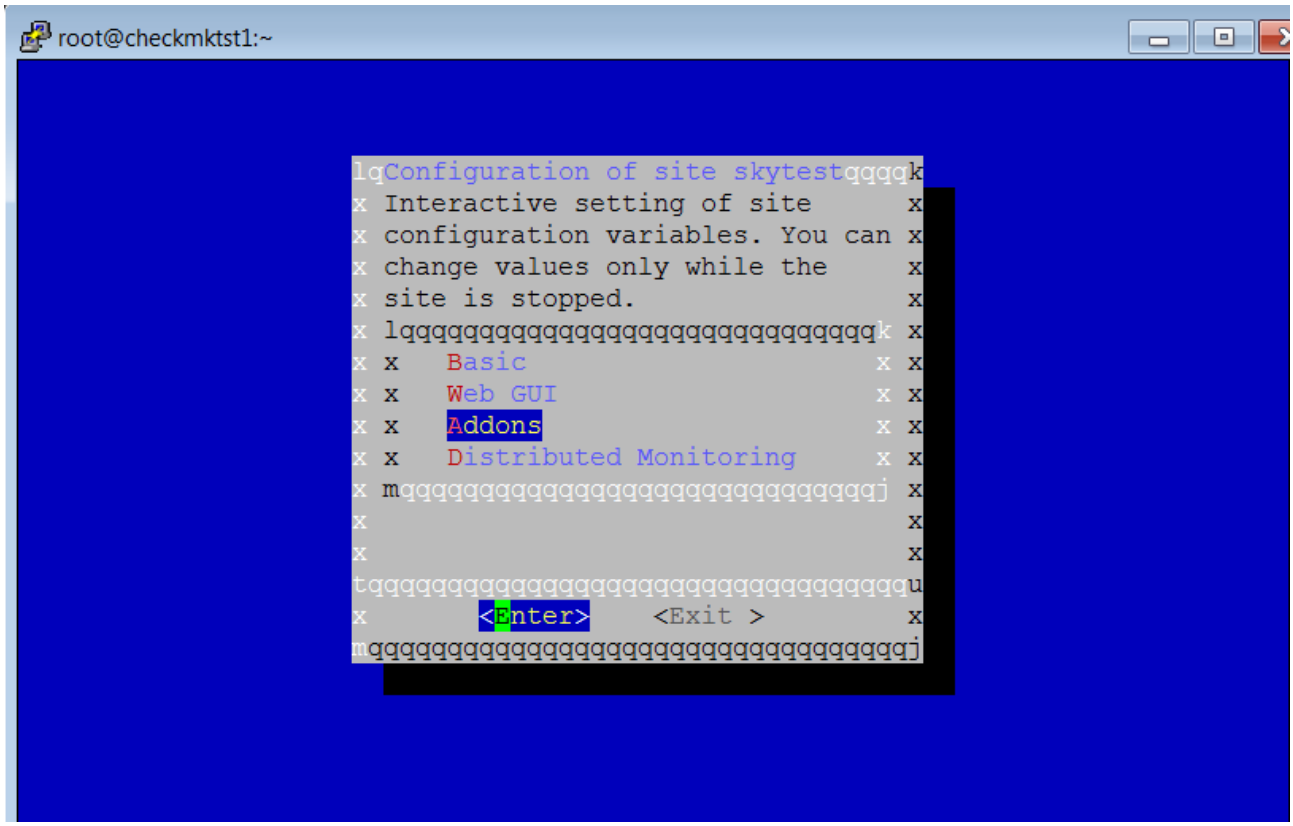
-receive incoming traps
-do a regex or filtering if necessary
-decide the level of criticality
-generate a service AUTOMATICALLY assigned to the monitored device
-AUTOCLEAR function meaning that if we receive an “OK” trap, the service should change from red (critical) to green (OK)

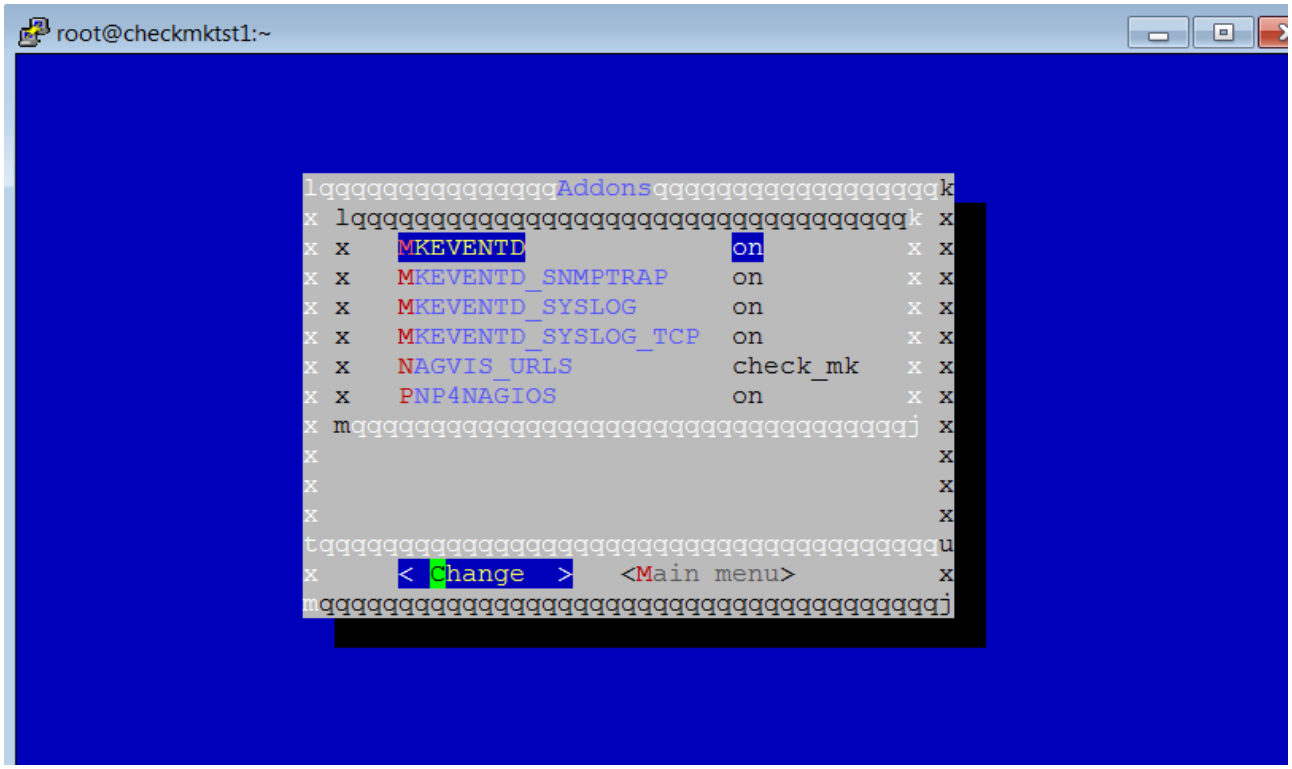
I’m going to list all the required steps but please note that I found the official documentation a little bit outdated and, depending on your environment (distribution as well Check_MK version and installation), some additional steps could be required.

- Connect to Check_MK host and, from the command line, run:

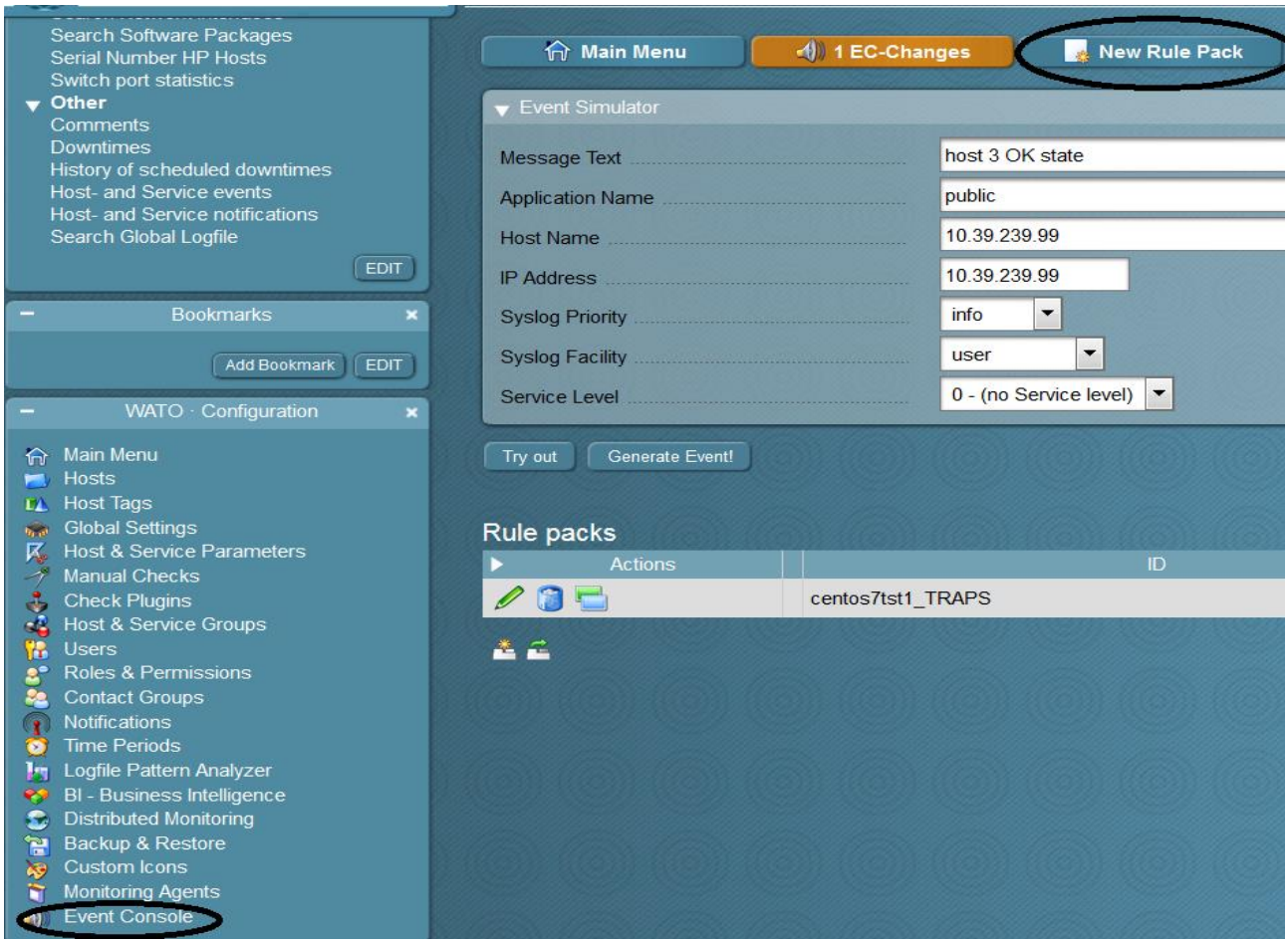
```
[root@checkmktst1 ~]# su - mysite
OMD[mysite]:~$ omd config
```

- Enable embedded *MKEVENTD_SNMPTRAP* and *MKEVENTD_SYSLOG*

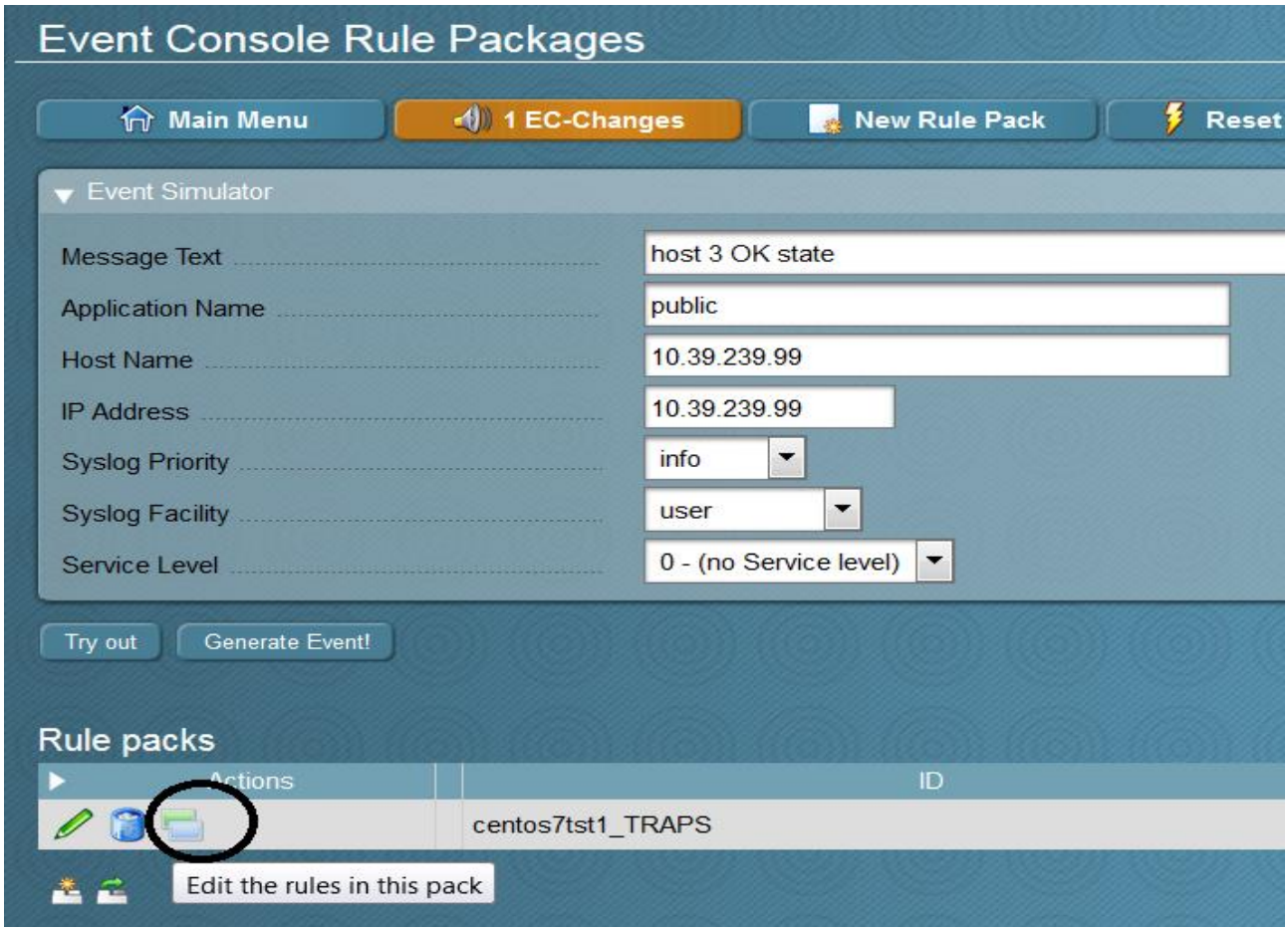




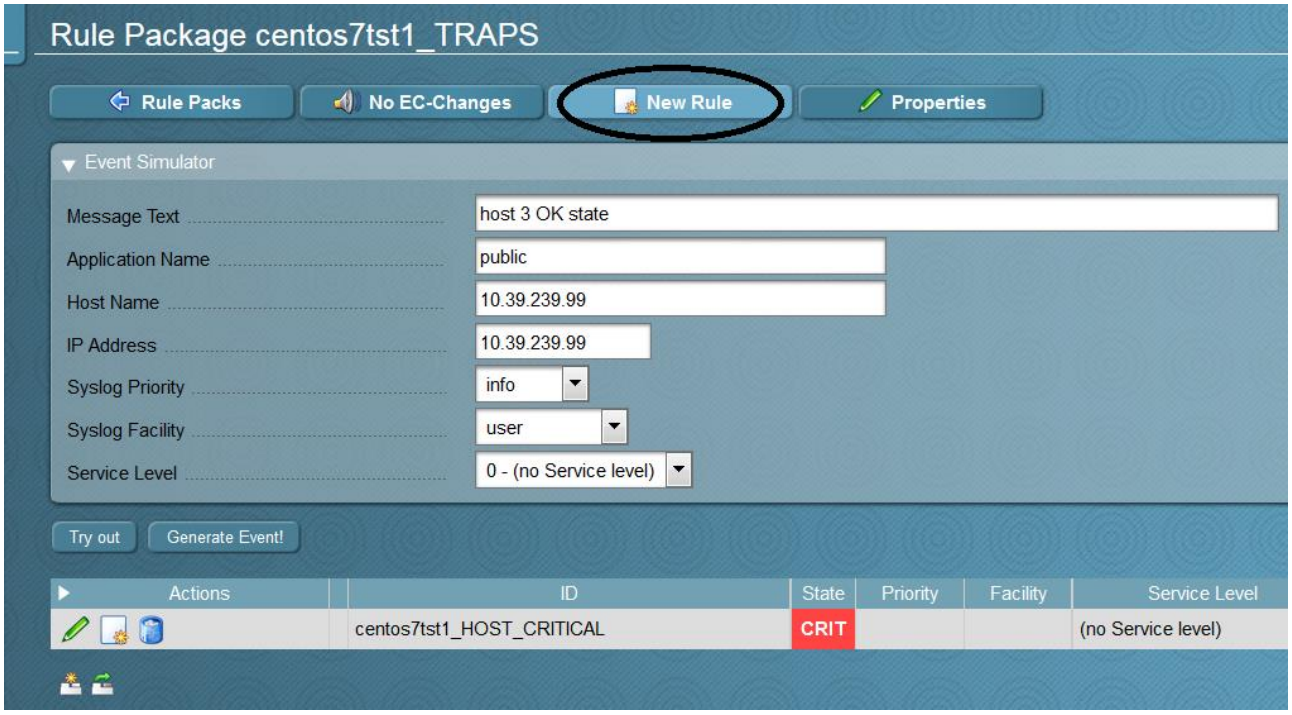
- *WATO-Configuration, Event Console, New Rule Pack*

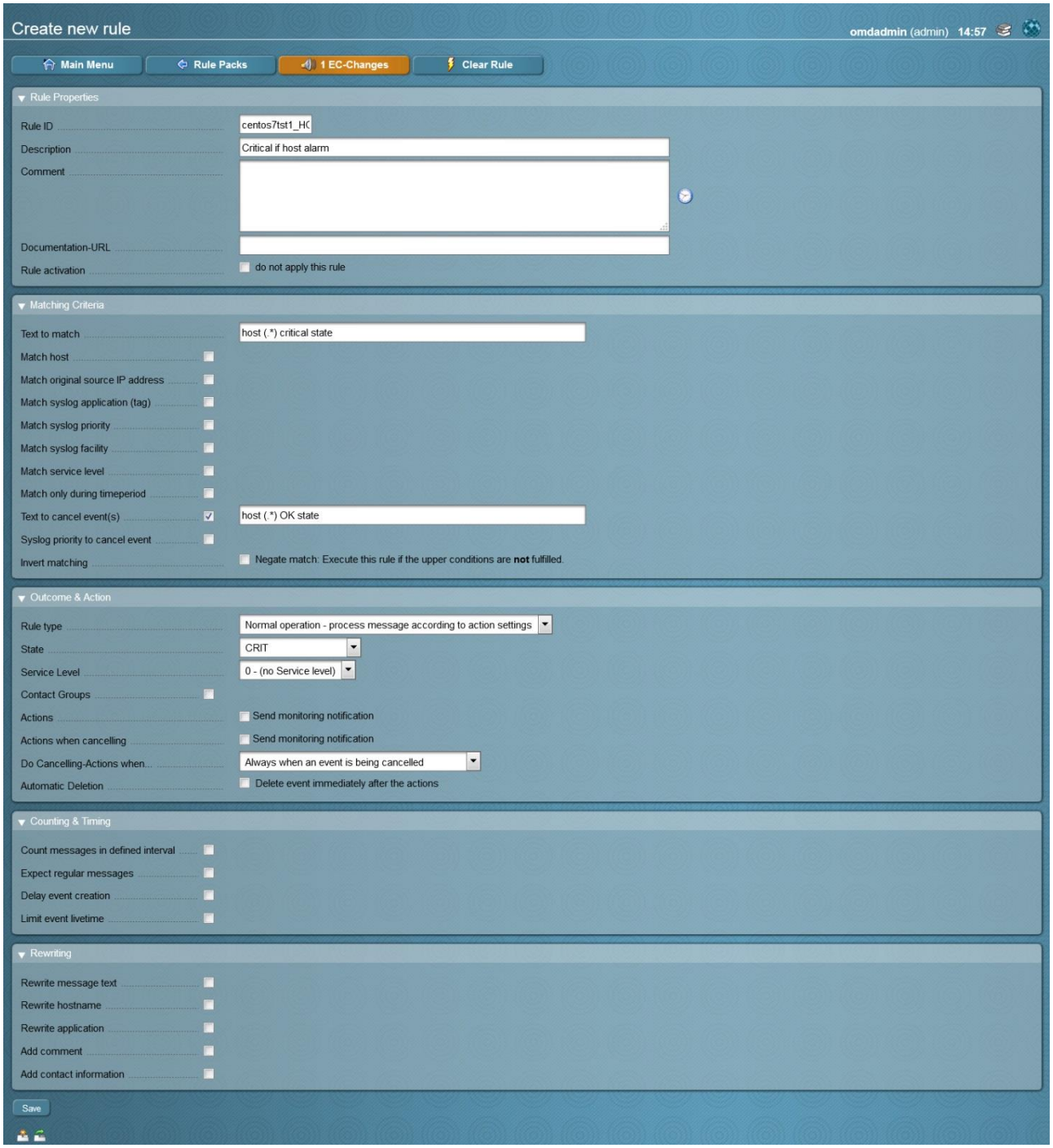


- Click the button *Edit the rules in this pack*

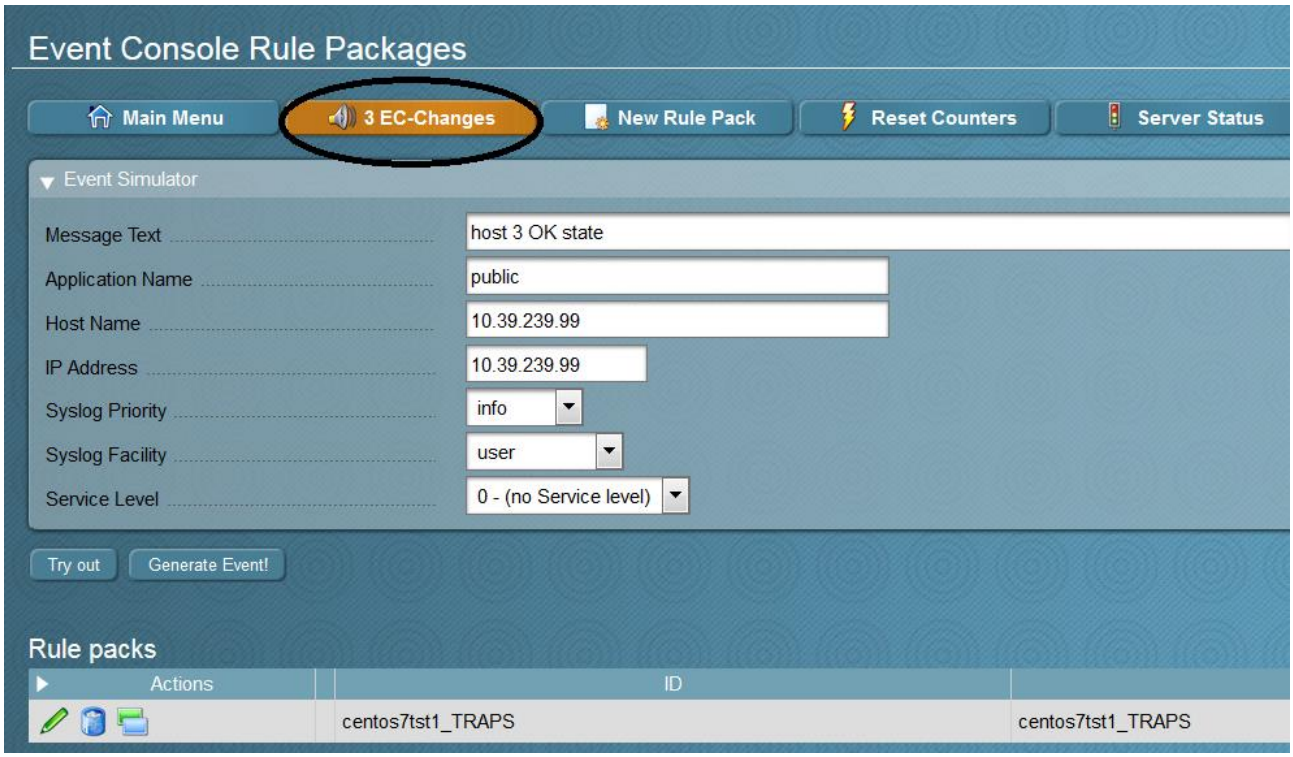


- Create a new rule like this




Create new rule
omdadmin (admin) 14:57
Main Menu
Rule Packs
1 EC-Changes
Clear Rule
Rule Properties
Rule ID
centos7tst1_HC
Description
Critical if host alarm
Comment
Documentation-URL
Rule activation
do not apply this rule
Matching Criteria
Text to match
host (.*) critical state
Match host
Match original source IP address
Match syslog application (tag)
Match syslog priority
Match syslog facility
Match service level
Match only during timeperiod
Text to cancel event(s)
host (.*) OK state
Syslog priority to cancel event
Invert matching
Negate match: Execute this rule if the upper conditions are not fulfilled.
Outcome & Action
Rule type
Normal operation - process message according to action settings
State
CRIT
Service Level
0 - (no Service level)
Contact Groups
Actions
Send monitoring notification
Actions when cancelling
Send monitoring notification
Do Cancelling-Actions when...
Always when an event is being cancelled
Automatic Deletion
Delete event immediately after the actions
Counting & Timing
Count messages in defined interval
Expect regular messages
Delay event creation
Limit event livetime
Rewriting
Rewrite message text
Rewrite hostname
Rewrite application
Add comment
Add contact information
Save

- Reload the configuration

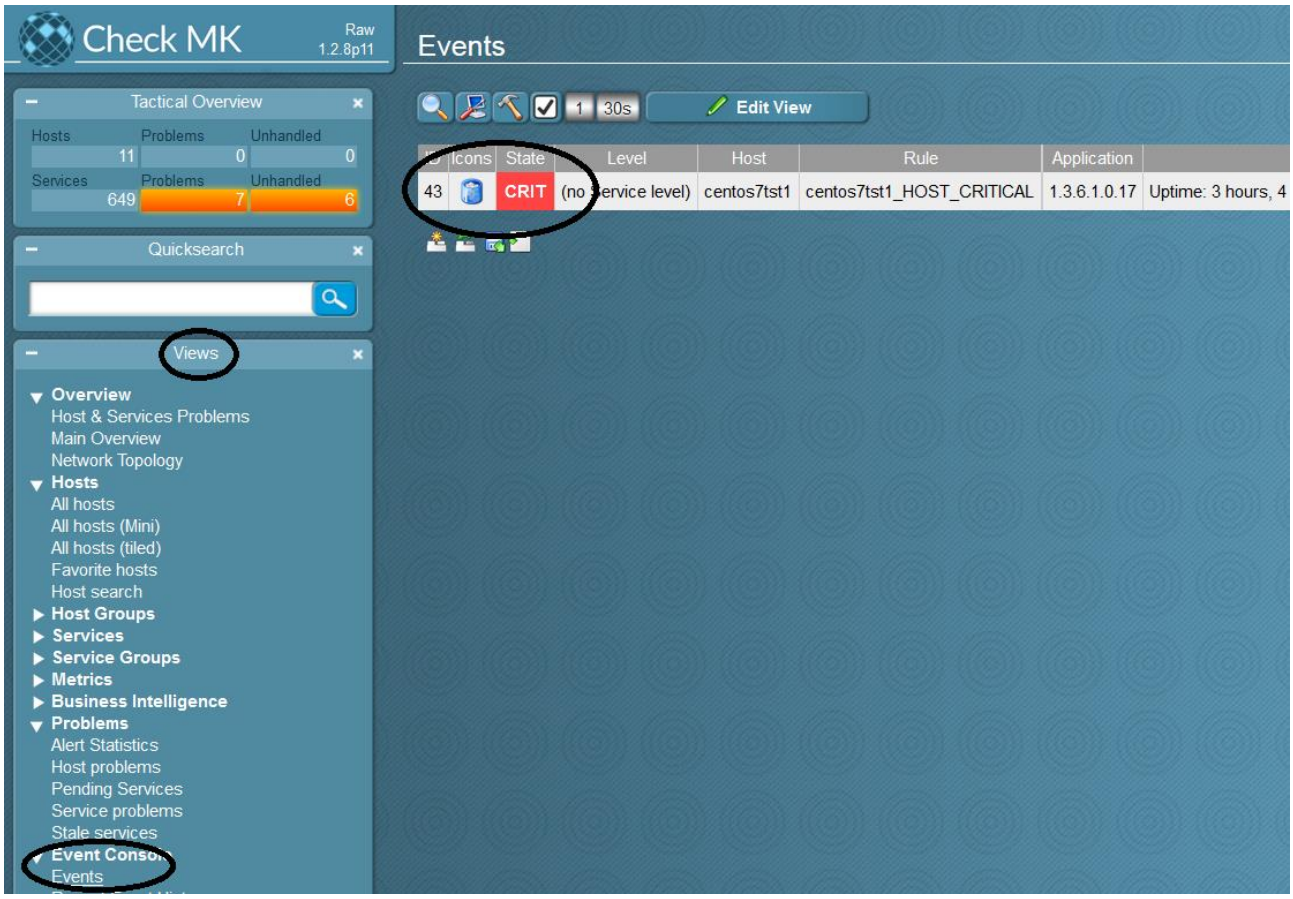


- Test the configuration

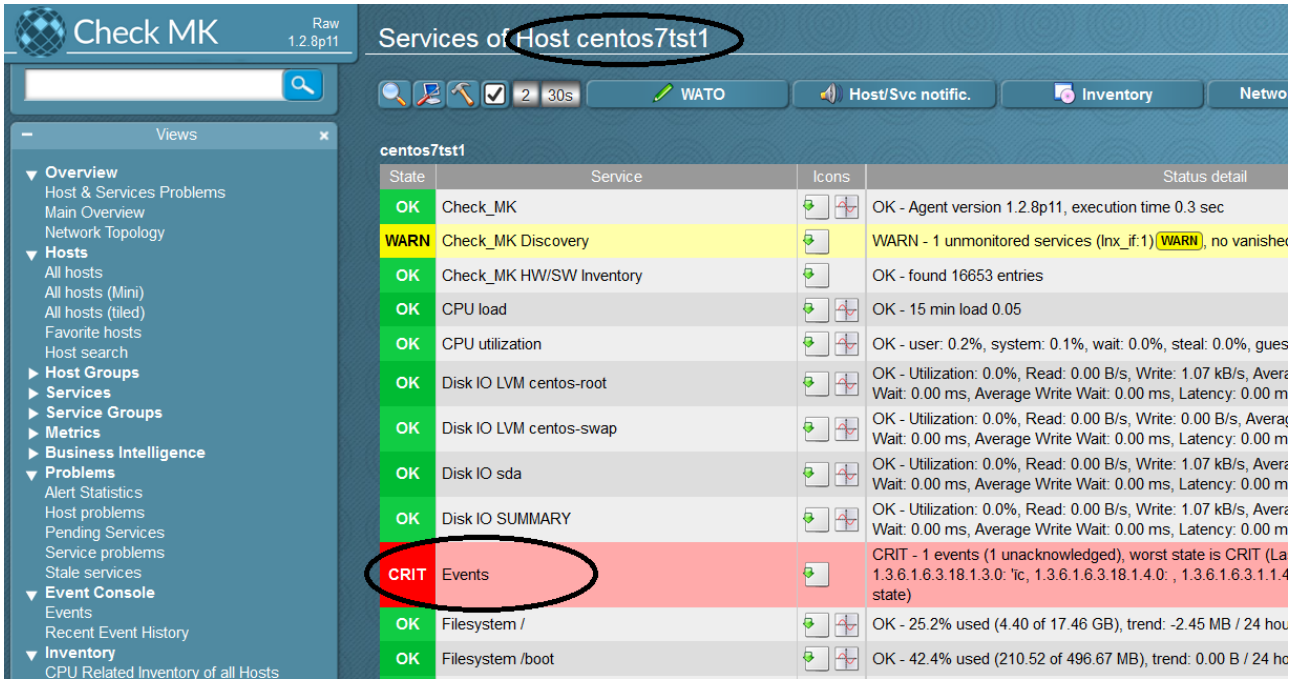
From the “centos7tst1” host run:

```
[root@centos7tst1 ~]# snmptrap -v 1 -c public 10.39.239.100 .1.3.6.1 10.39.239.99 6 17 ''
.1.3.6.1 s "host 3 critical state"
```

- Check if event has been created *WATO-View, Events*



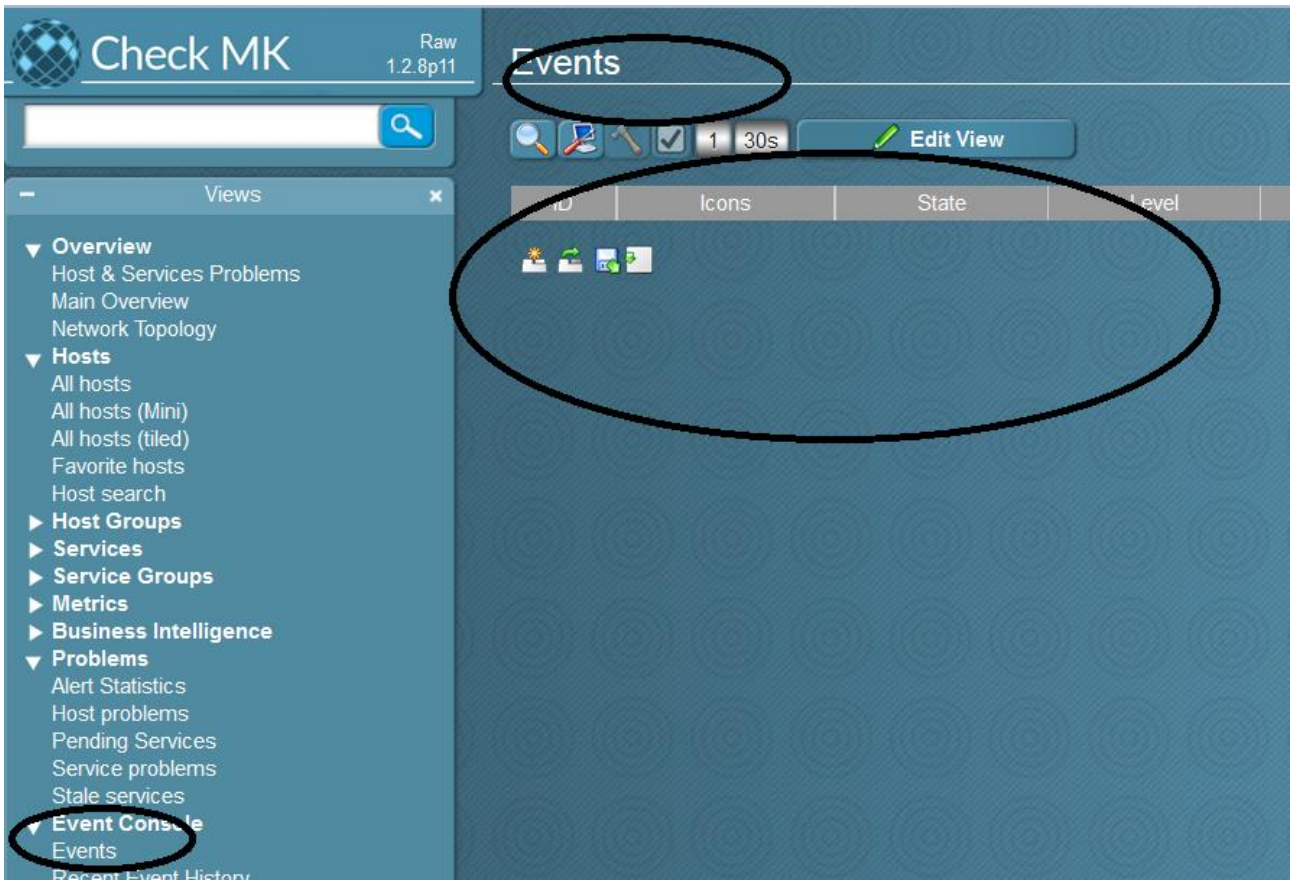
- Check that event has been AUTOMATICALLY associated to the correct host “centos7tst1”



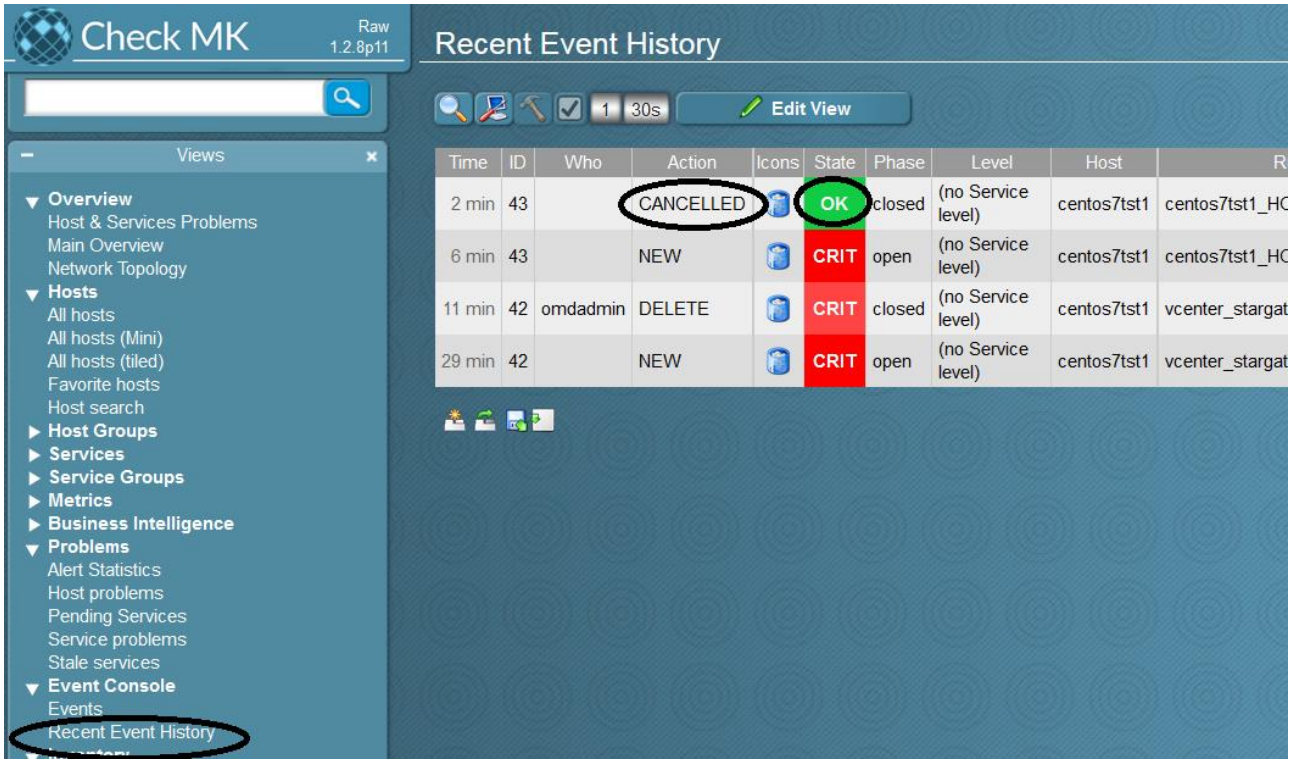
- Test the AUTOCLEAR mechanism is working correctly sending the following trap from the remote host

```
[root@centos7tst1 ~]# snmptrap -v 1 -c public 10.39.239.100 .1.3.6.1 10.39.239.99 6 17 ''
.1.3.6.1 s "host 3 OK state"
```

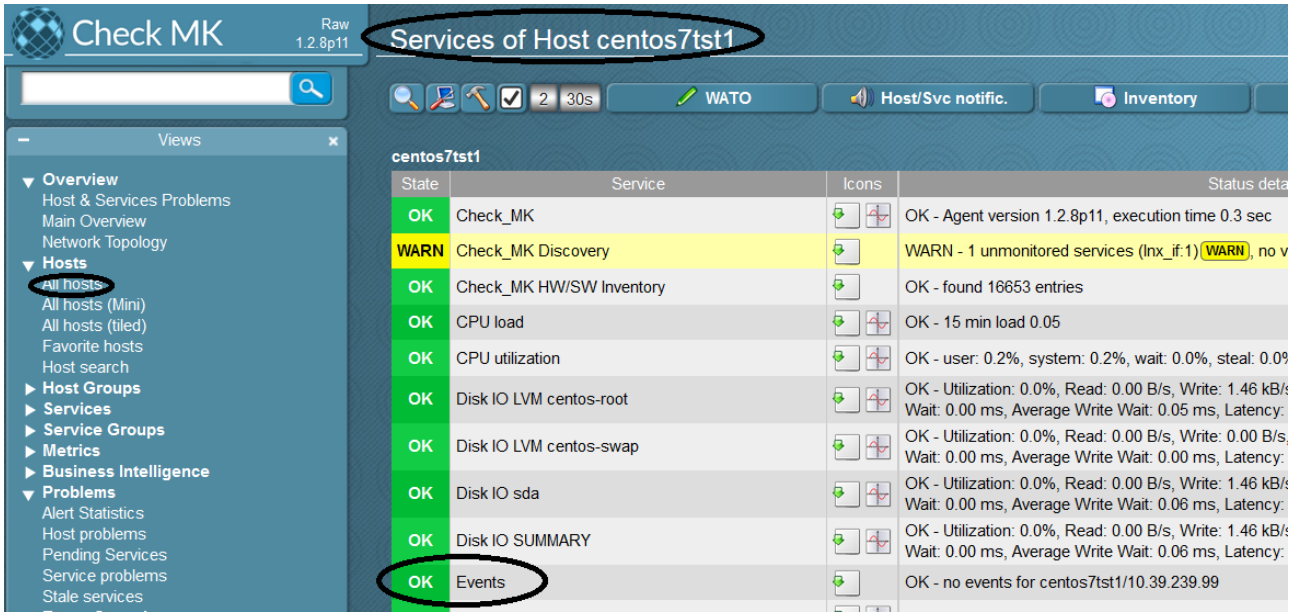
No open events should be displayed in *WATO-Views, Events*



But in *WATO-Views, Recent Event History* we can see that even the OK message has been received

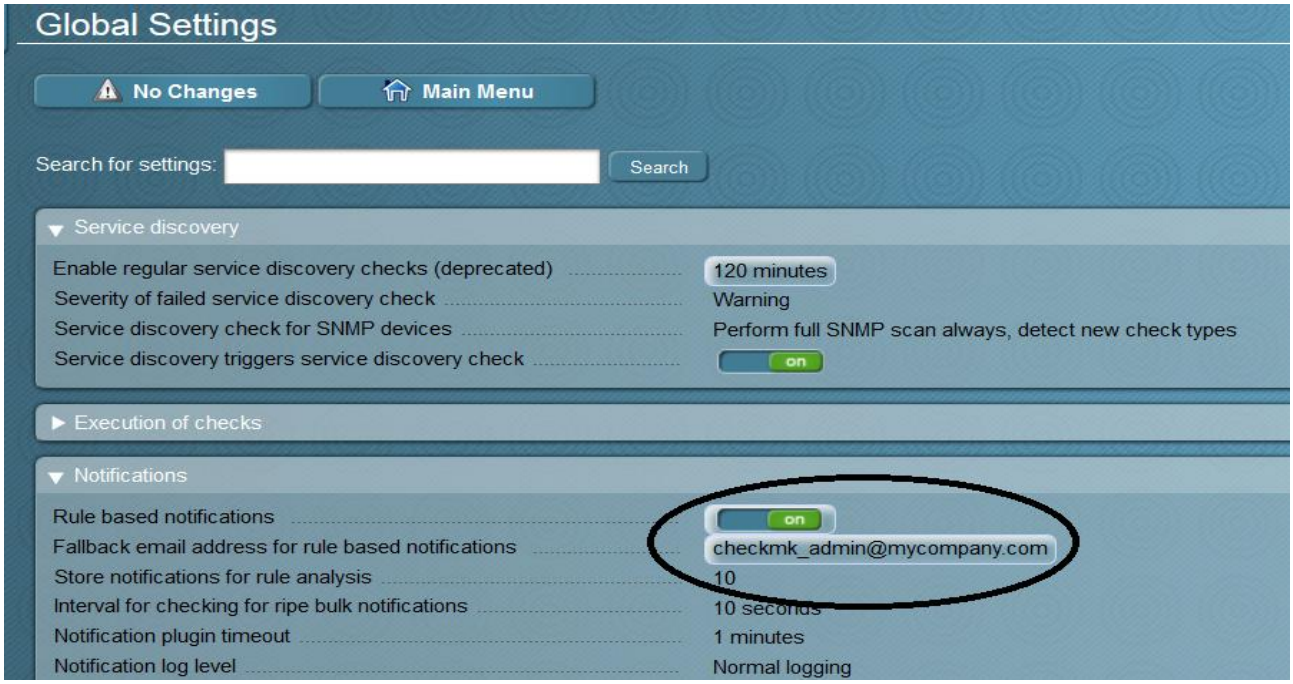


The *Event* service should be now green (OK)

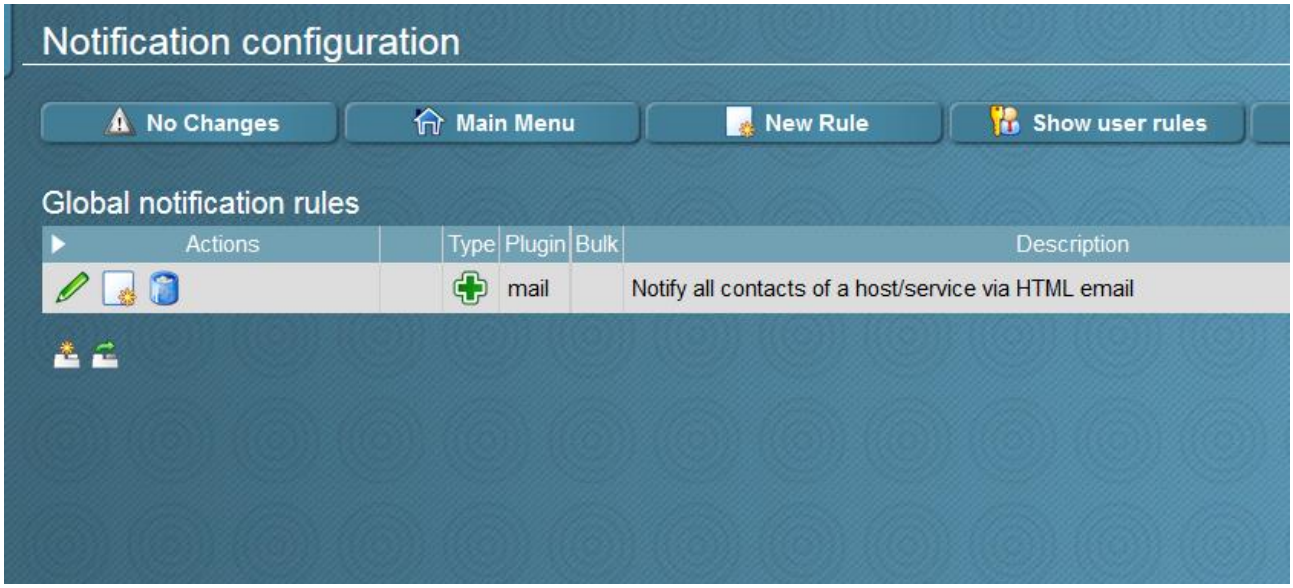


Managing Notifications

Notifications are quite a complex topic and Check_MK works very hard to make them as flexible as possible. Once again the best explanation of the thinking behind this comes from the Check_MK documentation: https://mathias-kettner.de/checkmk_rbn.html
Basically, notifications are managed using the new *RBN* (Rule Based Notifications) that add extra flexibility to the previous mechanism called *Flexible Notifications* by providing the separation of contact-assignment and notification.
The first step is to enable RBN and a failback address



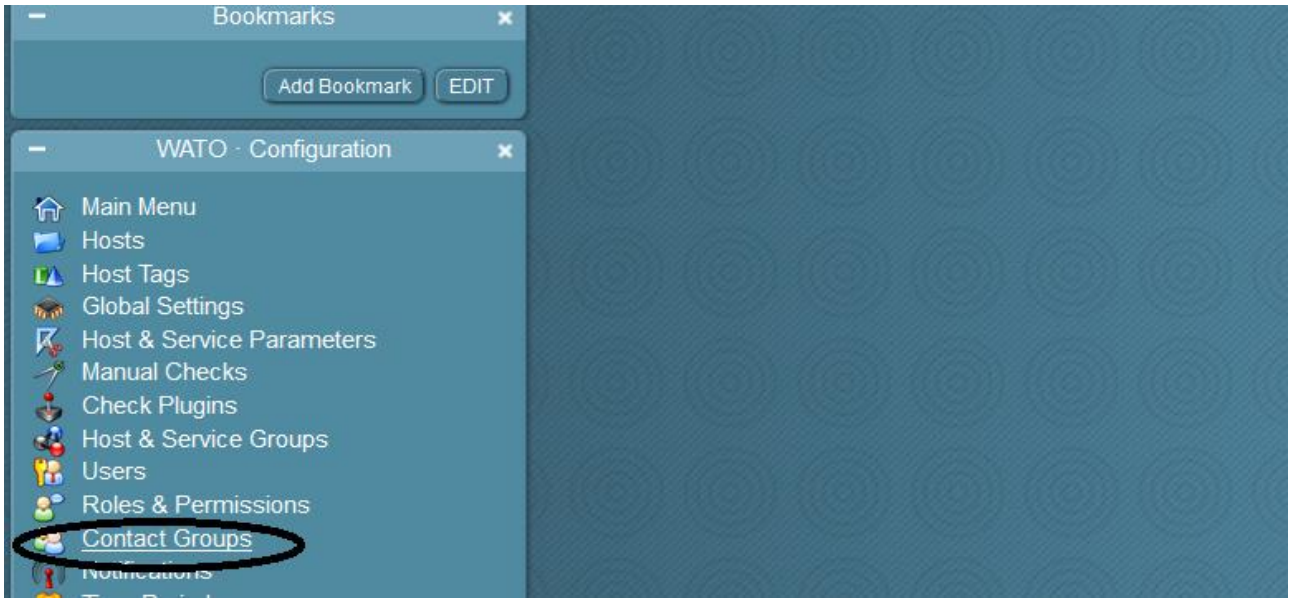
Now create a *Notification Rule* or change the exiting one*: WATO, Notifications*

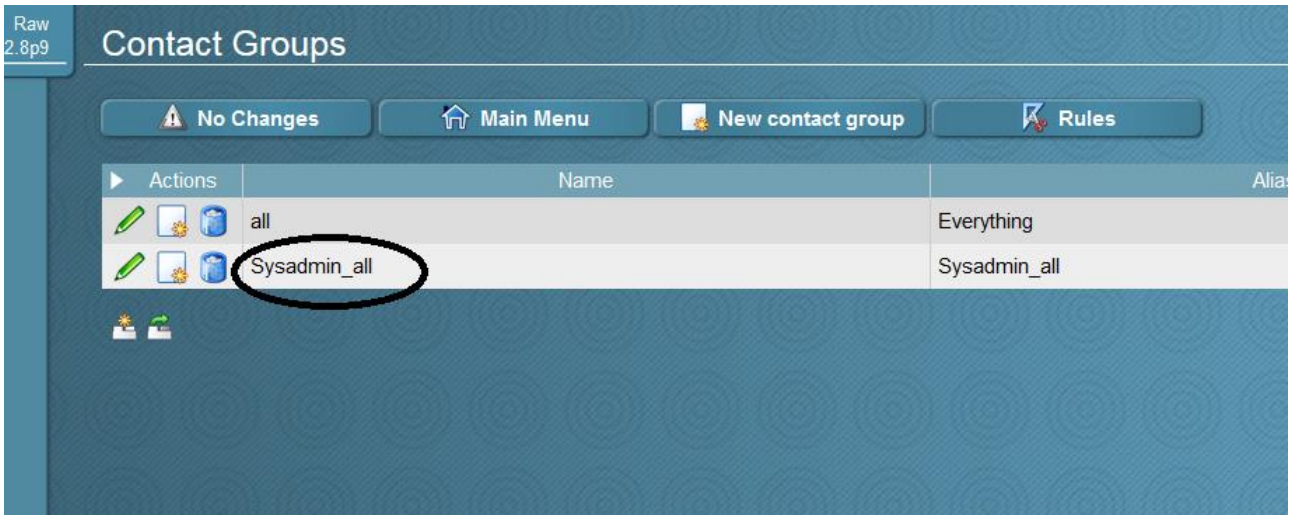


There are plenty of parameters that should satisfy all needs

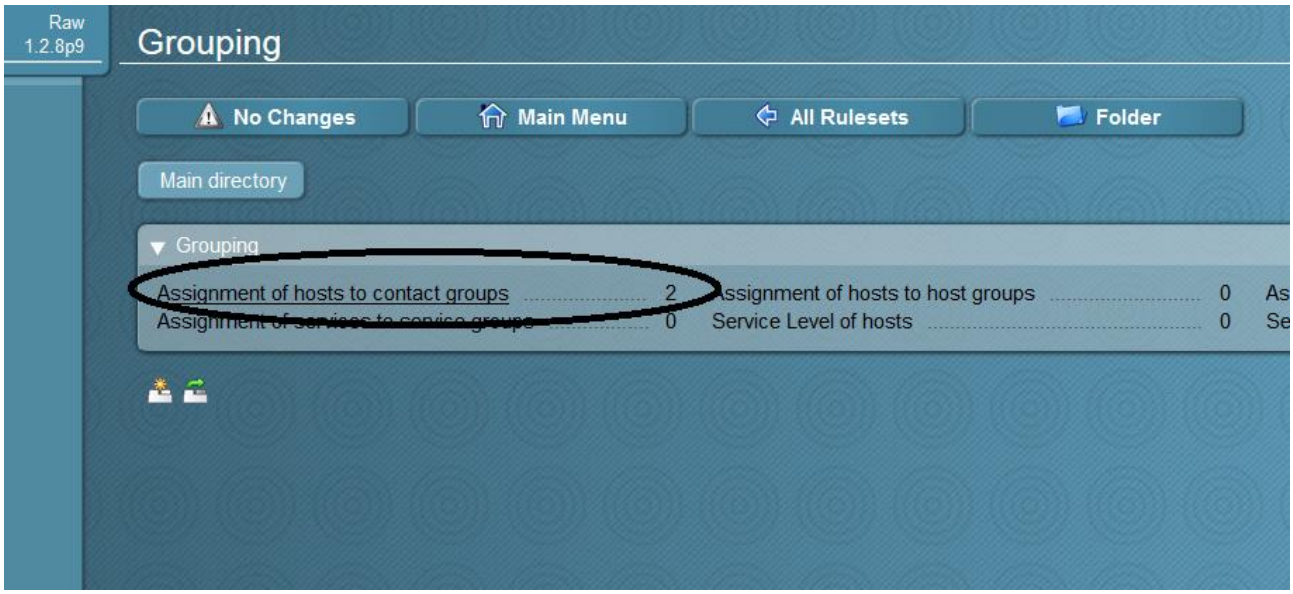
Contact group

Sometimes it is necessary to notify all people who are members of a specified contact group. This is done with the module *Contact Groups.* In this example, I created the *Sysadmin_all* contact group cloning the existing one called *all*

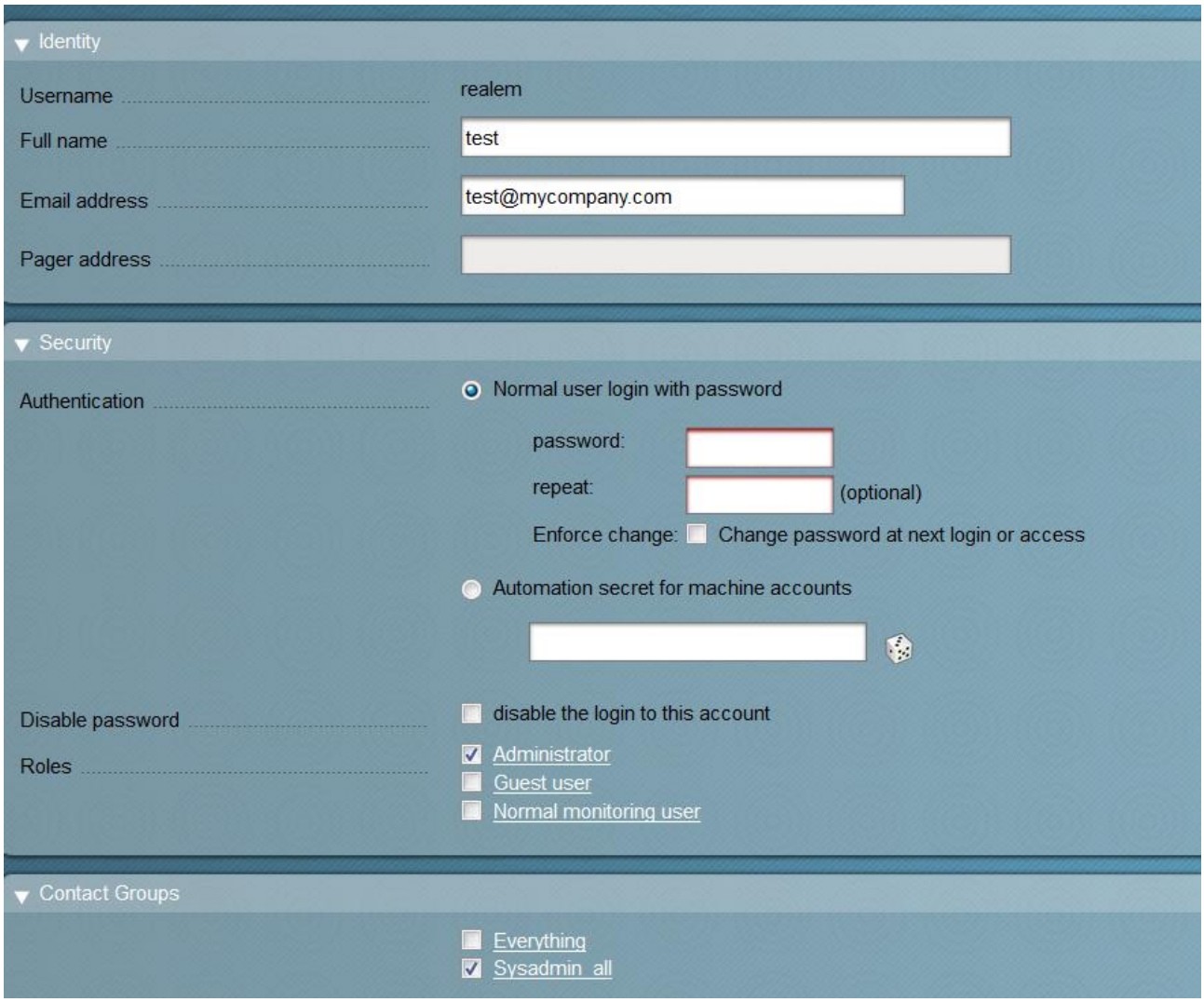




Important: put some hosts/services into that contact group. WATO: *Host & Service Parameters / Grouping / Assignment of hosts/services to contact groups*.



Create a user, enter an email address and put him into that contact group: WATO: *Users & Contacts*



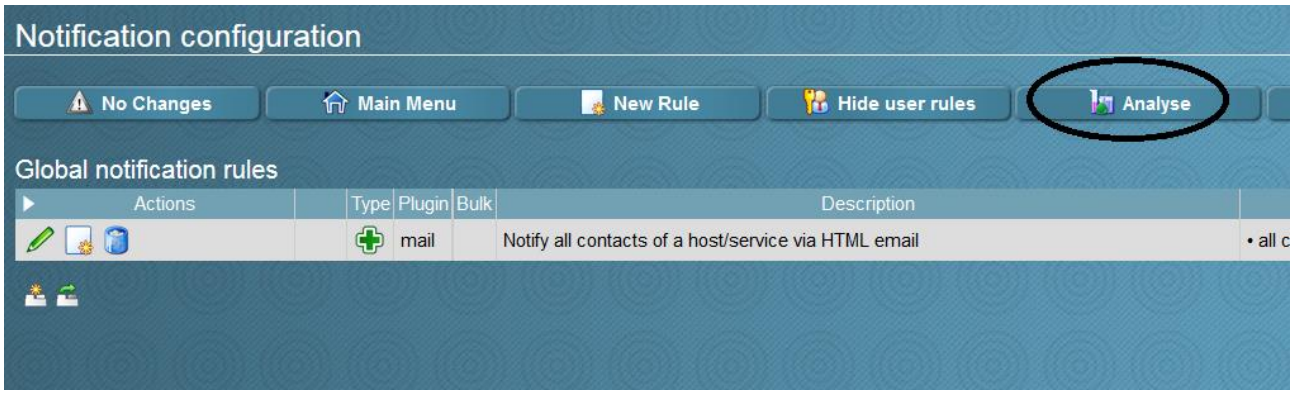
Activate Changes in WATO

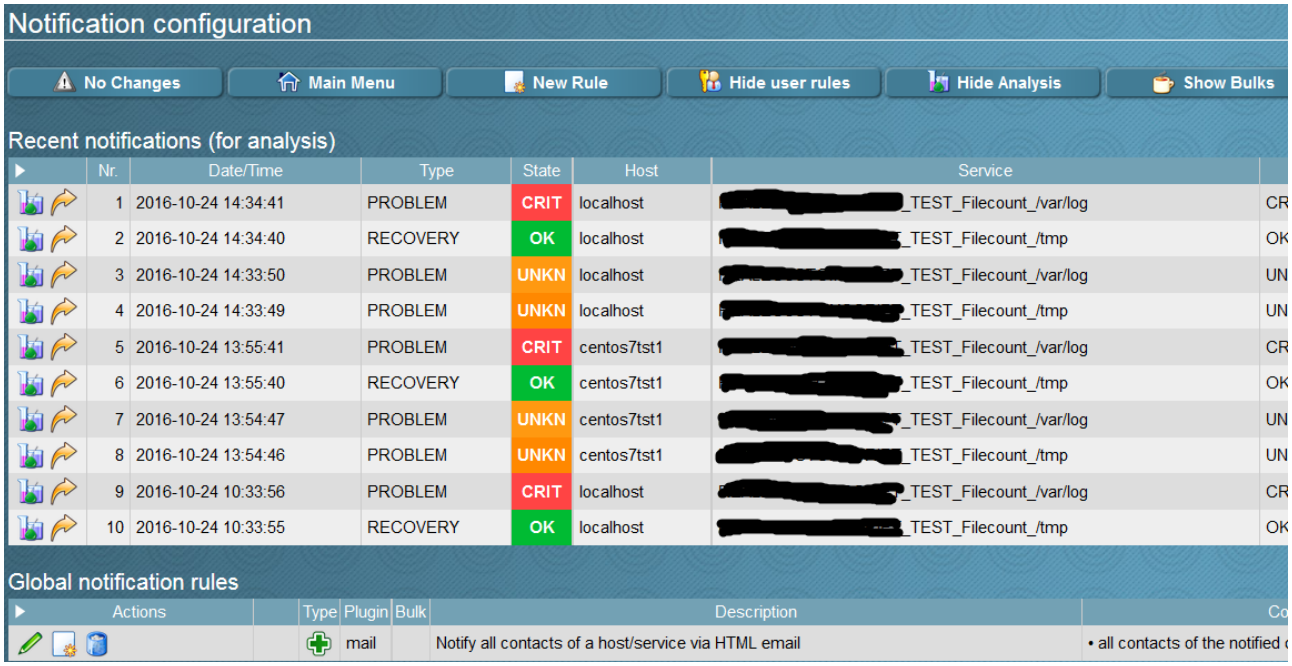
Analysis

To have alert notifications sent via email, make sure that your monitoring server is correctly setup so that it can send them. Test this with

```
echo "Mailbody" | mail -s "Testsubject" test@mycompany.com
```

If everything is setup properly, you should receive emails as soon as a *CRITICAL* service is detected. I also suggest to check the email log file, in my case */var/log/maillog* when troubleshooting this.
An *Analysis* tool is also available in the *Notifications Configuration* menu





Check_MK Update

The update process is generally very simple but, before proceeding, don't forget to take a backup and read the release notes very carefully. Problems could arise (especially with major upgrades) and it's good to have a backout process just in case.

Package installation

Download the latest package for your distribution and install it as shown:

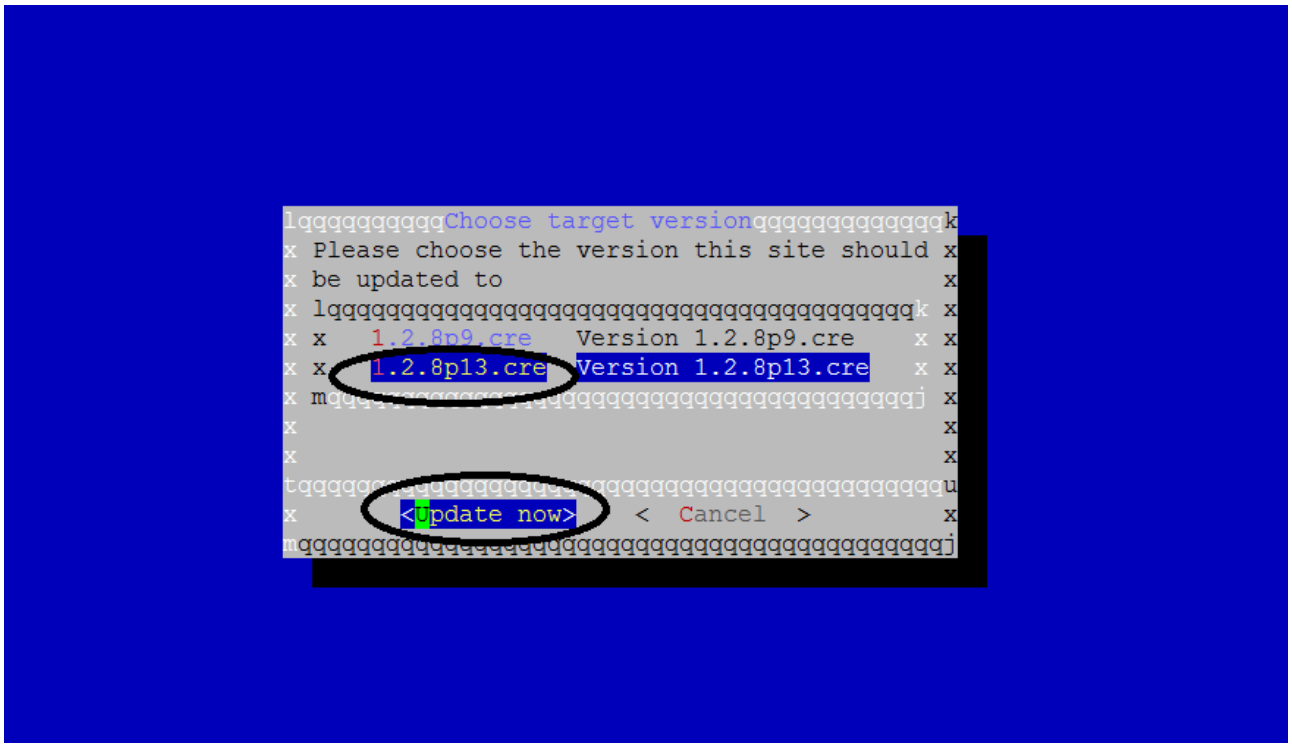
```
[root@checkmktst1 ~]# cd /tmp/
[root@checkmktst1 tmp]# wget https://mathias-kettner.de/support/1.2.8p13/check-m
k-raw-1.2.8p13-el7-36.x86_64.rpm
--2016-10-21 11:33:06--  https://mathias-kettner.de/support/1.2.8p13/check-mk-ra
w-1.2.8p13-el7-36.x86_64.rpm
Resolving mathias-kettner.de (mathias-kettner.de)... 178.248.246.154
Connecting to mathias-kettner.de (mathias-kettner.de)|178.248.246.154|:443... co
nnected.
HTTP request sent, awaiting response... 200 OK
```

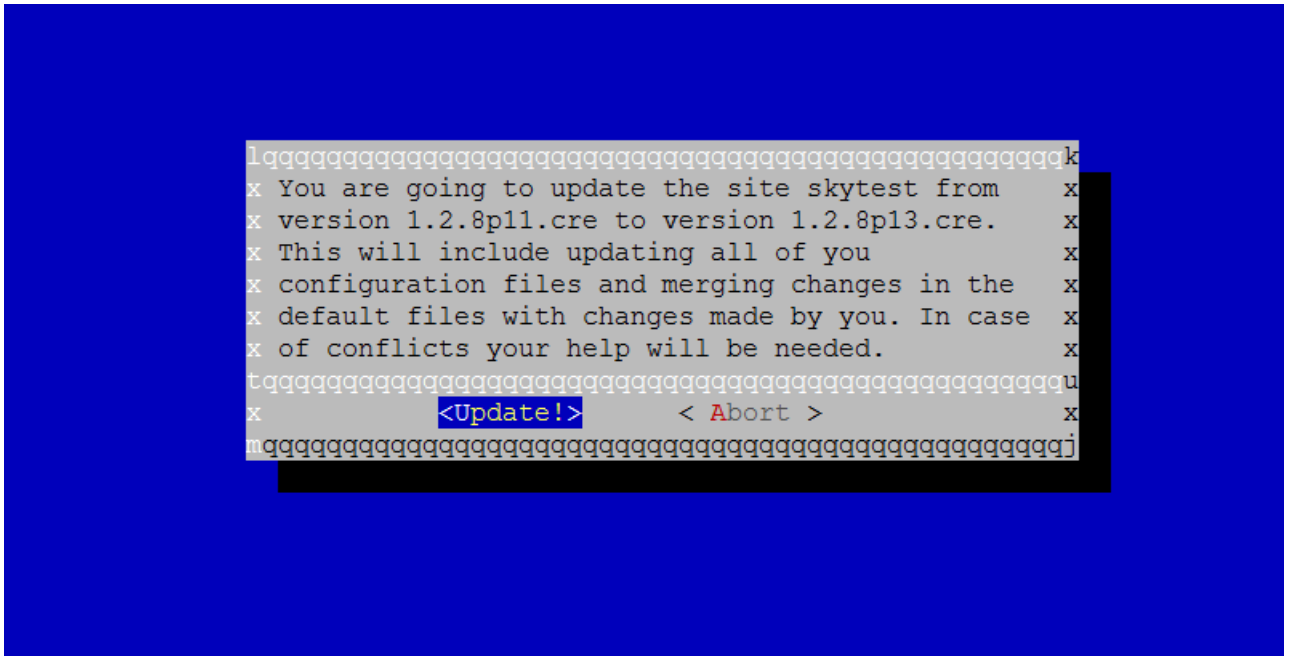
```
Length: 60640908 (58M) [application/x-redhat-package-manager]
Saving to: 'check-mk-raw-1.2.8p13-el7-36.x86_64.rpm'

91% [==================================>    ] 55,312,384   924KB/s   in 47s

2016-10-21 11:33:53 (1.13 MB/s) - Connection closed at byte 55312384. Retrying.

--2016-10-21 11:33:54--  (try: 2)  https://mathias-kettner.de/support/1.2.8p13/c
heck-mk-raw-1.2.8p13-el7-36.x86_64.rpm
Connecting to mathias-kettner.de (mathias-kettner.de)|178.248.246.154|:443... co
nnected.
HTTP request sent, awaiting response... 206 Partial Content
Length: 60640908 (58M), 5328524 (5.1M) remaining [application/x-redhat-package-m
anager]
Saving to: 'check-mk-raw-1.2.8p13-el7-36.x86_64.rpm'

100%[++++++++++++++++++++++++++++++++++++===>] 60,640,908  1.76MB/s   in 2.9s

2016-10-21 11:33:57 (1.76 MB/s) - 'check-mk-raw-1.2.8p13-el7-36.x86_64.rpm' save
d [60640908/60640908]

[root@checkmktst1 tmp]# rpm -Uvh check-mk-raw-1.2.8p13-el7-36.x86_64.rpm
Preparing...                          ################################# [100%]
Updating / installing...
   1:check-mk-raw-1.2.8p13-el7-36     ################################# [100%]
New default version is 1.2.8p13.cre.
```

Switching to the new version

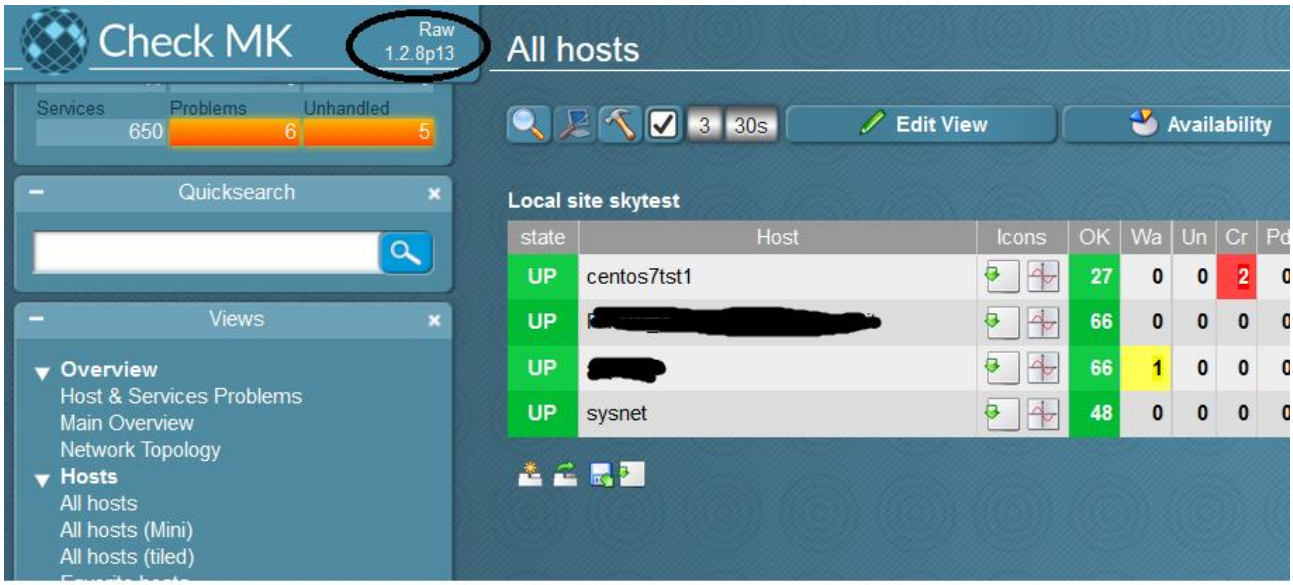
Switch to the new version using the OMD command:

```
[root@checkmktst1 tmp]# su - mysite
Last login: Thu Oct 20 16:59:39 CEST 2016 on pts/0
OMD[mysite]:~$ omd stop
Removing Crontab...OK
Stopping dedicated Apache for site mysite.....OK
Stopping nagios.....OK
Stopping npcd...OK
Stopping rrdcached...waiting for termination...OK
Stopping mkeventd...killing 15658......OK
OMD[mysite]:~$ omd update
```






```
OMD[mysite]:~$ omd start
```



Conclusion

I don't claim that check_MK is the best existing monitoring tool simply because I didn't tried all existing products but I can safely say that it is the best I have ever used. I have been also impressed by their clearness about the product's price: no complicated licensing model that force you to contact some sales manager (but they are willing to help you and know very well the product). In my opinion the price of the Enterprise Edition is ridiculous compared to other products and provides some nice additional features (apart the support) that it's worth a try.

The documentation is also very good even there are room for improvements and I have been able to monitor lot of enterprise class devices in few days without any headache thank also to the check_MK mailing list that is very active.