# MONITORING VMWARE-BASED CLOUD COMPUTING INFRASTRUCTURE WITH NAGIOS

Dariusz Sosnowski, Krzysztof Bielawski

Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

**Abstract:** Cloud computing (CC) is a very popular model of service. It provides clients with dynamic infrastructure or platform to perform various tasks. Monitoring of underlying infrastructure enables providers to assure a certain level of quality of service. This article describes requirements and methodology of deploying monitoring system based on Nagios, for CC infrastructure based on VMware virtualization software. It also presents a case study of such system.

**Keywords:** monitoring, VMware, Nagios, cloud computing

## 1. Introduction

### 1.1 Cloud computing model

Cloud computing (CC) is defined [1] as a service that provides on-demand network access to a pool of configurable computing resources. Depending on level of abstraction, the end user has access to defined instances of software or to whole infrastructure. Basically such functionality is implemented using multiple virtualization solutions, either commercial (e.g. VMware vSphere, Microsoft Hyper-V) or open-source (Linux KVM, Xen).

In available articles (e.g [2], [3]) CC infrastructure is divided into several layers. For the purpose of this paper, simplified model is presented:

– **Hardware and network infrastructure** - lowest layer in the model; physical servers, network infrastructure (routers, firewall, switches), storage arrays are representative elements of this layer; those entities provide resources (computing power - CPU time, RAM memory, disk space, etc.) that will be shared with the clients;

Advances in Computer Science Research, vol. 12, pp. 79-93, 2015.

– **Virtualization hypervisor** - virtualization systems and software used for management; enables service providers to distribute desired resources to clients, leveraging virtual machines and virtualized networks;
– **Virtual machines, operating system layer** - virtual machines created by entities in hypervisor layer, which purpose is to distribute computing resources to client; in this layer we also locate operating systems installed on those machines that are managed by provider or client (in Infrastructure as a Service - IaaS model);
– **Application** - software or platform accesible by cloud's end users; this layer is provided by service provider (Platform as a Service - PaaS or Infrastructure as a Service - IaaS model) or it is independent from provider (as in IaaS model, where client manages virtual machines and applications installed by himself);

In this paper focus is mostly put on IaaS model. This model is characterized by its separation of concerns. Clients manage ordered virtual machines and other resources by themselves. It means that service provider works only in **Hardware and network** and **Hypervisor** layer. Additionally provider might be working in **Operating system** layer, because some of the management servers can be virtualized.

### 1.2 Monitoring cloud computing infrastructure

From the provider perspective it is very important to have a reliable and robust monitoring system. It helps provider to assure a certain level of quality of service. Having such enables infrastructure administrators to quickly react on critical events and diagnose their causes. Information gathered and presented by monitoring system, can be assigned to certain layers of CC model:

– **Hardware and network infrastructure**
  • servers accesibility on the network
  • network traffic
  • temperature of CPU cores on virtualization hosts
  • power usage and load of server's power supplies and *power distribution units* (PDUs)
– **Virtualization hypervisor**
  • virtualization hosts' accessibility
  • virtualization hosts' memory and CPU usage
  • usage of disk storage supplied for storing virtual machines images
– **Virtual machines, operating system layer**
  • virtual machines' accessibility
  • virtual machines' memory and CPU usage

  - disk usage
  - checking if certain processes are working
- **Application** - maintained by the clients

There are automated solutions provided by virtualization software providers. VMware developed software named vCenter Operations Manager (vCOps) [4], which provides overview of hypervisor layer - state of virtual machines, resource usage of virtualization hosts. There are two major drawbacks of this solution. Firstly, it introduces addtional licensing cost for service provider. While it is not a problem for bigger corporations, still eliminating those costs might reduce price of the service for clients and in the case of smaller provider companies reduce operating costs. Secondly, its monitoring scope is limited to hypervisor layer. It does not allow us to inspect operating systems installed on virtual machines, review state of particular applications or monitor hardware specific metrics of underlying bare metal infrastructure. Goal of this article is to present a monitoring system, that eliminates those two disadvantages. It must be capable of integrating into CC infrastructure based on VMware virtualization stack and give access to operating system layer, what vCOps prevents us from. Eliminating licensing costs can be achieved by using open-source components.

In this article a monitoring system based on Nagios [5] is presented. Following sections provide more insight into Nagios and methods used to integrate Nagios with the infrastructure. Also a case study of the monitoring system built on top of Nagios and integrated into CC infrastructure is described.

## 2. Nagios monitoring software

Nagios is an open-source monitoring software, that is well recognized in professional community. This system can be used to monitor a low-level hardware and network infrastructure, but also high-level entities such as operating systems, web servers and other services.

Internally, an infrastructure is defined as a set of objects. *Host* objects represent any network-accessible hardware. *Host* objects have assigned multiple *Service* objects. Every *Service* object is responsible for one metric, service, application or any other property of the *Host*. State of *Service* is determined by the output of assigned *Command*, which represent a plugin to be run to pull *Host*/*Service* state. To every *Service* and *Host*, a *Contact* objects can be assigned, which hold contact information.

Following sections present features of Nagios, useful for monitoring CC infrastructure.

### 2.1 Plugin system

Nagios uses plugins to inspect a state of servers and applications. Initial set of plugins contains programs such as:

- **check_disk** - check local host's disk usage
- **check_swap** - check local host's swap usage
- **check_snmp** - perform an SNMP query and compare the result to given numeric thresholds or regular expressions
- **check_http** - perform an HTTP(S) request to given server and compare output and output to given pattern
- **check_by_ssh** - runs a command on remote server using a SSH protocol, and uses that command output

System administrator can easily create his own plugins. It can be written in any programming language and Nagios uses only its standard output and return code. Return code is used to determine host/service status. Standard output is used for textual representation of this state. This paper presents technologies and methods that can be used to create plugins for Nagios, to integrate monitoring system into VMware virtualization infrastructure. Because of Nagios plugin system's flexibility it can also be included in many others CC infrastructures.

### 2.2 Active and passive checks

By default Nagios supports active state checking. In this pattern, a monitoring system establishes a connection with certain host and pulls necessary information to determine service state. Another pattern is passive checking, where parts of the system send their state to Nagios. Nagios has implemented this paradigm and it can be enabled for subset or whole infrastructure. Using this paradigm in monitoring system can lead to increased performance of the system, by reducing load on monitoring server.

### 2.3 Notifications

In case of critical states of host or service, system administrators must be informed. Nagios has implemented simple system of notifications. When host or service changes its state, a predefined command is executed. That command can send an email, SMS, etc. - it depends on configuration. It has access to host/service name, state, plugin output and contact information.

### 2.4 Event handlers

For every host or service state change, system administrator can define an optional command to be run. It allows monitoring system to be an active member of infrastrcture, by reacting to certain events. This way Nagios can restart failed services, log events to database, enter ticket to helpdesk center, etc.. This feature is optional, but using it can provide much more sophisticated system.

## 3. Nagios integration with cloud computing infrastructure based on VMware stack

### 3.1 Hardware and network layer

Hardware and network layer of CC infrastructure includes:

- physical servers that provide computing resources, i.e. CPU and RAM
- storage arrays (specialized servers capable of managing multiple hard drives, usually with RAID support)
- network switches, routers and firewalls
- power distribution units (PDU)

Aside from monitoring tools provided by manufacturers of these devices, most of them implement *Simple Network Management Protocol* (SNMP) [7]. It is a protocol defined by *Internet Engineering Task Force* (IETF). SNMP exposes management data in form of variables with explicitly defined *object identifiers* (OID). OIDs are sequences of numbers, which represent namespaces of variables and specific variables. For example OID 1.3.6.1.2.1.3 represents variable which holds system uptime.

SNMP allows to use *Management Information Bases* (MIBs), which provide textual representations of OIDs, e.g. OID for sysUpTime variable is defined in RFC1213-MIB MIB and can be referred as RFC1213-MIB::sysUpTime. Every manufacturer can support the set of standard modified MIBs (IF-MIB for network interfaces, HOST-RESOURCES-MIB for physical server resources) and provide MIBs specialized for their hardware (Dell provides IDRAC-MIB-SMIv2 to access management information for its blade servers).

SNMP by default uses UDP protocol on port 161. Port 162 is used by monitoring server, to receive *traps*. Traps are hardware-generated (or more generally - client-generated) SNMP messages, that indicate critical events happening in the system.

There are standard utilities for Linux systems to manage servers that accept SNMP requests. On Debian-based distributions *snmpget* and *snmpwalk* tools can be

used for sending GET requests. OIDs or MIB variables can be used for SNMP objects identification. Nagios has *check_snmp* plugin (written in C) packaged with standard plugins. It can be defined to pull certain OIDs from given host.

In monitoring hardware and network layer in cloud computing infrastructure IF-MIB information base can be used. It is defined to access hardware interfaces data such as interface description, bytes received, bytes sent, MTU, etc.. All those values are located in IF-MIB::ifTable table. Every row (IF-MIB::ifEntry objects) of this table represents one interfaces of the device and contains multiple variables including:

- ifDescr - interface name (possibly defined by operating system)
- ifInOctets - counts bytes received
- ifOutOctets - counts bytes sent

Those variables can be used to monitor traffic on interfaces of physical servers and switches. Usage of this data, along with hardware-specfic MIBs is presented in the case study.

### 3.2 Hypervisor layer

Hypervisor layer, as noted in introduction, consists of virtualization systems that enable provider to distribute resources to clients, using virtual machines. Main part of VMware virtualization stack is vSphere ESXi [8] operating system, that acts as hypervisor - software used for creation and management of virtual machines. Hypervisor is installed on every physical server (referred later as *host*). In VMware systems an underlying disk storage is abstracted for hypervisors as *Datastores*. Datastore provides unified interface for vSphere ESXi to store virtual machines images, and can wrap multiple storage backends (e.g. NFS, iSCSI, FibreChannel). Every host, datastore, virtualized network can be controlled by vCenter server that should be configured in infrastructure.

vCenter server exposes HTTP API known as *vSphere API* [9] that can be used to pull state of hypervisor layer. It is designed as set of managed objects and each represents a single element of virtualization infrastructure. Those objects provide information about structure and functioning of infrastructure. VMware provides bindings for Ruby language - RbVmomi [11]. This paper presents methods how to get information about hypervisor hosts and datastores state and use it to create Nagios plugins.

RbVmomi is constructed as a direct mapping of vSphere API objects structure. Code snippet below presents how to pull hypervisor host (describes by IP address) memory usage:

```
1 conn = RbVmomi::VIM.connect(host: ip,
2                              user: username,
3                              password: password,
4                              insecure: true)
5 datacenter = conn.serviceInstance
6                  .find_datacenter('Datacenter')
7 host = datacenter.hostFolder
8                  .findByIp(host, RbVmomi::VIM::HostSystem)
9 perfManager = conn.servinceInstance.content.perfManager
10 memory_max_mb = host.hardware.memorySize / (1024 * 1024)
11 memory_used_mb = perfManager.retrieve_stats([host],
12                                              ['mem.consumed'])
13 usage = (memory_used_mb / memory_max_mb) * 100.0
```

In the **1st** line a connection to vCenter server is established by using *connect* method of *VIM* class. Following - a *Datacenter* object is pulled in the **5th** line. *Datacenter* represents a group of hosts, datastores, networks, etc. which usually are located in one physical place. This object encapsulates objects of other entities. In the **7th** line *HostSystem* object is pulled to variable *host*, which represents a single vSphere ESXi host in infrastructure, identified by its IP address in datacenter. *Host* object is used to gather maximum memory of this host (in **10th** line, value is hold in bytes). *PerformanceManager* object is used to gather various metrics of the system. It used to retrieve *host's* consumed memory (metric names as *'mem.consumed'*) in **11th** line. Finally, the percentage usage of host's memory is calculated (line **13**).

Host's CPU usage can be accessed nearly the same way, using alternative metric in query to *PerformanceManager* object - *'cpu.usagemhz'*. Full list of metrics is accesible in [10].

Datastore usage can be accessed similarly. The following code snippet presents the usage of RbVmomi to access given datastore usage data (datastores are recognized by their name).

```
1 conn = RbVmomi::VIM.connect(host: ip,
2                              user: username,
3                              password: password,
4                              insecure: true)
5 datacenter = conn.serviceInstance
6                  .find_datacenter('Datacenter')
7 datastore = datacenter.find_datastore(datastore_name)
8 summary = datastore.summary
9 capacity, free = summary.capacity, summary.free
10 used_bytes = capacity - free
11 usage = (used_bytes / capacity) * 100.0
```

Beginning of the code is a standard procedure for establishing the connection with vCenter server. Then, *Datastore* object is pulled from *Datacenter* (line **7**). *DatastoreSummary* object, which is *Datastore*'s property, encapsulated metrics of the queried datastore. Overall capacity and free space is pulled from this object (line **9**) and percent of usage is calculated.

This methods can be used to create Nagios plugins. Additionally some user-defined thresholds can be applied (for warning and critical states) to further improve a quality of information presented to administrators.

### 3.3 Operating system layer

Operating system layer consists of virtual machines and specific operating systems installed on them. In this paper we want to monitor some set of machines (with Linux-based or Windows operating system) that are managed by service provider. Those machines are used to manage service resources, infrastructure and serve as backend for website or used by clients to loan resources.

**Linux-based systems** Nagios provides preferred solution for monitoring Linux-based systems. Nagios Remote Plugin Executor (NRPE) [6] is a software consisting of two parts - plugin for Nagios server, and daemon for monitored server. Daemon must be placed on monitored server. NRPE daemon accepts connection over SSL or SSH and executes commands issued by Nagios server. It is configured to accept only specified set of commands. Nagios server issues execution of plugins, by using *check_nrpe* plugin.

**Windows systems** Every newer Windows system have builtin system for monitoring. Windows Management Instrumentation (WMI) [12] is the infrastructure for accessing management data of running Windows machine. System administrators are provided with SQL-like language for accessing operating system metrics, such as:

- CPU and memory usage
- running processes list
- disk storage

Those informations can be accessed both locally and remotely. *wmic* [13] is a Linux implementation of WMI client and can be used by Nagios plugins to monitor Windows machines.

### 3.4 Gathering performance metrics

Nagios by itself does not store metrics returned by plugins. It can be configured to execute certain command to save plugin outputs or sent it to other systems. Although, preferable solution is to use system designed for gathering, storing and presenting metrics. Cacti [14] is an example of such system. It allows us to store metrics in Round Robin Databases (RRD) [15] which are design for storing time-series data. By default Cacti uses SNMP protocol to gather information, but it can also use user created plugins. Plugins developed for Nagios can be used for this task, with slight modification of output format. For more information, refer to documentation found at [14].

## 4. Case study

In this section, the case study of developed monitoring system is presented. Description is provided, by dividing aspects of the system like it was described in introduction.

### 4.1 Hardware and network layer

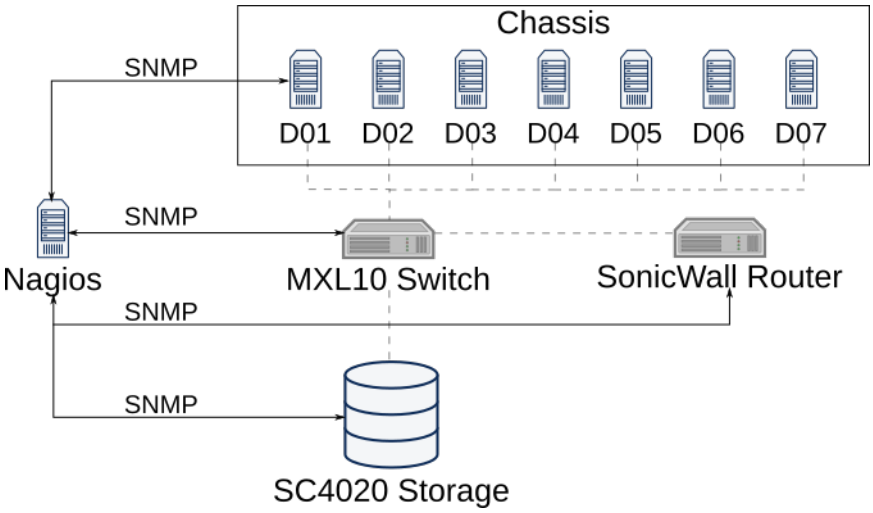


**Fig. 1.** Hardware layer in the case study

Hardware layer is presented in Figure 1. It consists of several Dell M620 servers, connected to Dell M1000e chassis which provide power and networking for servers. Servers and chassis are connected to Force MXL10 switch. Dell SC4020 storage array is also connected to the switch. SonicWall router is responsible for routing network traffic and acts as firewall.

Interest is put on monitoring accessibility of all those elements in network and network traffic on MXL10 switch and SonicWall router. Temperature of CPU cores of blade servers, temperature inside chassis and its power usage are also needed to monitor.

Accessibility is checked by using ICMP ping standard, realised in *check_ping* Nagios plugin. As noted in section 3.1 network traffic can be monitored by using variables degined in IF-MIB information base for SNMP protocol.

Dell provides system administrators with *iDRAC-MIB-SMIv2* information base for its blade servers, including M620 servers. This MIB holds table named *temperatureProbeTable*, which provides readings from temperature probes located near CPU cores. Every row of table consists of fields like, *temperatureProbeLocationName* - probe location and *temperatureProbeReading* - probe reading.

For Dell M1000e *DELL-RAC-MIB* information base can be used. This base holds table named *drsCMCPSUTable* which contains data of chassis power supplies. Every row of this table contains fields like *drsCMCPSULocation* - location of the probe, *drsCMCPSUAmpsReading* - amperage of power supply unit and *drsCMCPSUVoltsReading* - voltage on power supply. Fields *drsCMCAmbientTemperature*, *drsCMCProcessorTemperature* and *drsChassisFrontPanelAmbientTemperature* defined in MIB can be used to pull respectively CMC ambient temperature (CMC stands for Chassis Management Controller), internal CMC processor and temperature of chassis front panel.

| Host | Service | Status | Last Check | Duration | Attempt | Status Information |
|---|---|---|---|---|---|---|
| Dell M1000e | CMC ambient temperature | OK | 04-05-2015 12:40:02 | 0d 0h 1m 2s | 1/3 | SNMP OK - CMC ambient temperature 40 |
| | CMC processor temperature | OK | 04-05-2015 12:40:01 | 0d 0h 1m 3s | 1/3 | SNMP OK - CMC processor temperature 39 |
| | Front panel temperature | OK | 04-05-2015 12:39:19 | 0d 0h 1m 45s | 1/3 | SNMP OK - Front panel temperature 26 |
| | PING | OK | 04-05-2015 12:39:15 | 64d 9h 48m 40s | 1/3 | PING OK - Packet loss = 0%, RTA = 0.92 ms |
| | PS-1 amps reading | OK | 04-05-2015 12:39:33 | 0d 1h 22m 31s | 1/3 | SNMP OK - PS-1 amps reading 0.817 |
| | PS-2 amps reading | OK | 04-05-2015 12:39:39 | 0d 1h 21m 21s | 1/3 | SNMP OK - PS-2 amps reading 1.25 |
| | PS-3 amps reading | OK | 04-05-2015 12:39:27 | 0d 1h 21m 37s | 1/3 | SNMP OK - PS-3 amps reading 0.938 |
| | PS-4 amps reading | OK | 04-05-2015 12:39:39 | 0d 1h 21m 20s | 1/3 | SNMP OK - PS-4 amps reading 0.939 |
| | PS-5 amps reading | OK | 04-05-2015 12:39:59 | 0d 1h 21m 55s | 1/3 | SNMP OK - PS-5 amps reading 1.637 |
| | PS-6 amps reading | OK | 04-05-2015 12:39:41 | 0d 1h 21m 19s | 1/3 | SNMP OK - PS-6 amps reading 0.751 |
| Dell M620 slot 01 | CPU1 Temperature | OK | 04-05-2015 12:39:26 | 0d 1h 35m 37s | 1/3 | SNMP OK - CPU1 temperature 480 |
| | CPU2 Temperature | OK | 04-05-2015 12:39:47 | 0d 1h 34m 4s | 1/3 | SNMP OK - CPU2 temperature 470 |
| | PING | OK | 04-05-2015 12:39:50 | 24d 8h 54m 47s | 1/3 | PING OK - Packet loss = 0%, RTA = 0.70 ms |
| Dell M620 slot 02 | CPU1 Temperature | OK | 04-05-2015 12:39:22 | 0d 1h 31m 41s | 1/3 | SNMP OK - CPU1 temperature 460 |
| | CPU2 Temperature | OK | 04-05-2015 12:40:01 | 0d 1h 31m 51s | 1/3 | SNMP OK - CPU2 temperature 530 |
| | PING | OK | 04-05-2015 12:39:27 | 64d 9h 48m 44s | 1/3 | PING OK - Packet loss = 0%, RTA = 0.92 ms |

**Fig. 2.** Excerpt from Nagios view of hardware layer

*check_snmp* plugin can be utilized to monitor network traffic and temperatures in physical hardware of the hardware layer. Accessing hardware and network layer data through SNMP layer, provides provider with more insight into infrastructure state than using only vCOps. Because of its limitation to inspect only hypervisor layer, even simple SNMP monitoring used in this Nagios-based systems extends monitoring scope, eliminating one of the disadvantages of the system.

### 4.2 Hypervisor layer

Hypervisor layer is presented in Figure 3. It consists of several hosts with vSphere ESXi hypervisor installed, vCenter server used for management of those hypervisors and set of VMware datastores named *MGMT-xx* and *PROD-xx*, where *xx* is an zero-padded number. Only way to monitor those entities from Nagios is to utilize vSphere API exposed by vCenter server. To integrate those two systems, plugins were developed, which used vSphere API Ruby implementation RbVmomi, highlighted in section 3.2. Utilising the methodology presented in this section, we developed plugins to check memory and CPU usage by vSphere ESXi hosts and usage of VMware datastores. Accessibility of vSphere ESXi hosts in the network is checked with *check_ping* plugin.

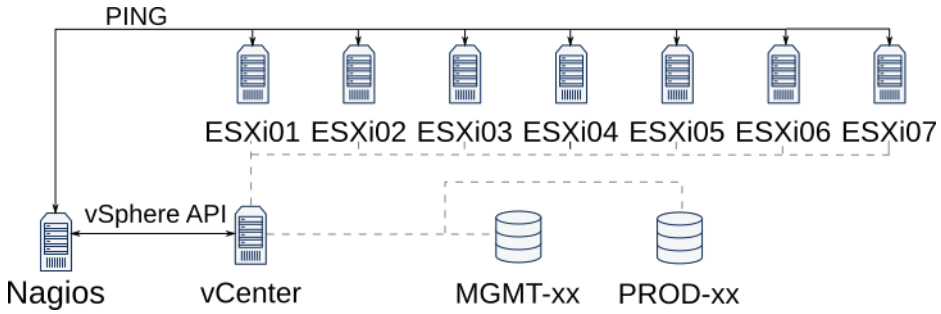


**Fig. 3.** Hypervisor layer in the case study

Despite certain advantages of this solution (simplicity of flow of data and certainty about data correctness), having such monitoring introduces a single point of failure. If vCenter server stops working (e.g. internal software error), whole hypervisor layer becomes invisible to hypervisor layer. Though this unwanted situation is by all means critical, it would cause a wave of events concerning unavailability of

| Host | Service | Status | Last Check | Duration | Attempt | Status Information |
|---|---|---|---|---|---|---|
| ESXi01 | CPU usage | OK | 04-05-2015 12:58:08 | 21d 8h 52m 2s | 1/3 | ESXi CPU OK: usage 6.14% |
| | Memory usage | OK | 04-05-2015 12:56:37 | 21d 8h 55m 16s | 1/3 | ESXi Memory OK: usage 58.24% |
| | PING | OK | 04-05-2015 12:59:01 | 66d 8h 41m 23s | 1/3 | PING OK - Packet loss = 0%, RTA = 0.35 ms |
| ESXi02 | CPU usage | OK | 04-05-2015 12:58:21 | 21d 8h 55m 19s | 1/3 | ESXi CPU OK: usage 3.52% |
| | Memory usage | OK | 04-05-2015 12:58:56 | 21d 8h 56m 18s | 1/3 | ESXi Memory OK: usage 56.13% |
| | PING | OK | 04-05-2015 12:58:32 | 66d 8h 42m 9s | 1/3 | PING OK - Packet loss = 0%, RTA = 0.35 ms |

**Fig. 4.** Excerpt from vSphere ESXi hosts state from Nagios

vSphere ESXi hosts, which is not the case. This certain payoff must be accepted because of the nature of VMware virtualization stack. It is closed source software and one cannot interfere into internal functioning of hypervisors. Also it is discouraged by professional community to place third-party software inside working hypervisors.

| | | | | | | |
|---|---|---|---|---|---|---|
| MGMT-vCenter-01 | MGMT-01-SC4020 datastore usage | OK | 05-05-2015 16:36:54 | 49d 14h 31m 14s | 1/3 | Datastore MGMT-01-SC4020 usage OK: usage 53.74% |
| | MGMT-02-SC4020 datastore usage | CRITICAL | 05-05-2015 16:37:25 | 0d 0h 0m 46s | 1/3 | Datastore MGMT-02-SC4020 usage CRITICAL: usage 71.00% |
| | PING | OK | 05-05-2015 16:37:50 | 67d 12h 20m 7s | 1/3 | PING OK - Packet loss = 0%, RTA = 0.46 ms |
| | PROD-01-SC4020 datastore usage | OK | 05-05-2015 16:36:44 | 42d 9h 16m 6s | 1/3 | Datastore PROD-01-SC4020 usage OK: usage 34.94% |
| | PROD-02-SC4020 datastore usage | OK | 05-05-2015 16:34:58 | 49d 14h 31m 48s | 1/3 | Datastore PROD-02-SC4020 usage OK: usage 40.07% |
| | PROD-03-SC4020 datastore usage | OK | 05-05-2015 16:35:01 | 42d 9h 16m 47s | 1/3 | Datastore PROD-03-SC4020 usage OK: usage 41.69% |
| | PROD-04-SC4020 datastore usage | OK | 05-05-2015 16:35:30 | 42d 9h 16m 9s | 1/3 | Datastore PROD-04-SC4020 usage OK: usage 43.33% |

**Fig. 5.** Excerpt from Datastores state from Nagios. Here error message about overusage of storage.

### 4.3 Operating system layer

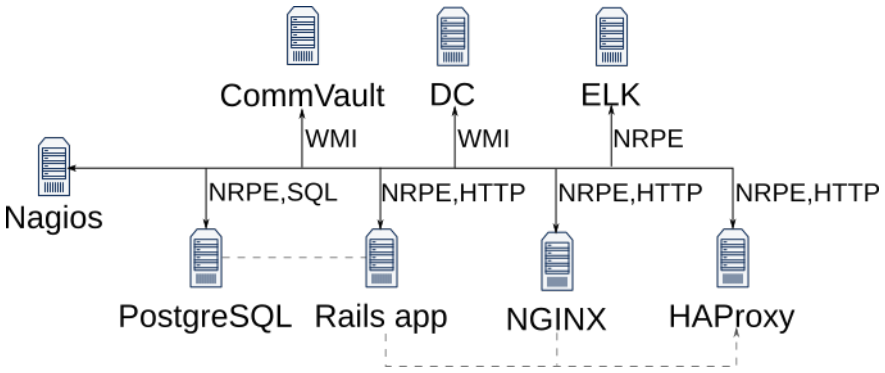


**Fig. 6.** Operating system layer in the case study

Operating system layer is presented in Figure 6. It consists of several hosts, including 2 virtual machines with Windows operating system *CommVault* - with backup

software and *DC* - domain controller for Active Directory. There are also several Debian hosts, i.e. server with PostgreSQL database, server with running Ruby on Rails application, server that servers static content through NGINX, server with HAProxy software (reverse proxy) and ELK server with Elasticsearch database for log gathering and Logstash for log receiving. Windows hosts are monitored using WMI infrastructure described in section 3.3 and utilising *Check WMI Plus* plugin [16], that uses *wmic* Linux client.

| | | | | | | |
|---|---|---|---|---|---|---|
| MGMT-DC-01 | CPU usage | OK | 05-05-2015 16:57:53 | 1d 2h 22m 44s | 1/3 | OK (Sample Period 177 sec) - Average CPU Utilisation 0.14% |
| | Drive C: usage | OK | 05-05-2015 16:55:03 | 56d 13h 56m 28s | 1/3 | OK - C: Total=99.66GB, Used=15.53GB (15.6%), Free=84.12GB (84.4%) |
| | Memory usage | OK | 05-05-2015 16:58:01 | 56d 13h 55m 13s | 1/3 | OK - Physical Memory: Total: 8GB - Used: 1.294GB (16%) - Free: 6.706GB (84%) |
| | PING | OK | 05-05-2015 16:59:12 | 43d 10h 5m 22s | 1/3 | PING OK - Packet loss = 0%, RTA = 0.73 ms |
| MGMT-ELK-01 | HTTP GET / | OK | 05-05-2015 16:51:04 | 14d 11h 7m 52s | 1/3 | HTTP OK: HTTP/1.1 200 OK - 2284 bytes in 0.002 second response time |
| | Logstash service | CRITICAL | 05-05-2015 16:59:45 | 0d 0h 0m 7s | 1/3 | LOGSTASH CRITICAL: not running |
| | Node elk01 document count | OK | 05-05-2015 16:59:26 | 14d 11h 13m 37s | 1/3 | Elasticsearch document count for node elk01 OK: 18872859 documents |
| | Node elk01 store size | OK | 05-05-2015 16:56:05 | 14d 14h 31m 43s | 1/3 | Elasticsearch storage for node elk01 OK: 7750.5 MB used |
| | PING | OK | 05-05-2015 16:58:49 | 14d 11h 16m 44s | 1/3 | PING OK - Packet loss = 0%, RTA = 0.56 ms |

**Fig. 7.** Excerpt from Operating System layer monitoring. Error message of Logstash service stopped working is shown.

Linux hosts are monitored for CPU load, memory usage and disk storage usage using Nagios basic plugins in conjunction with NRPE software. Additonally state of web services (Ruby on Rails application, NGINX web server and HAProxy reverse proxy) are checked issuing HTTP requests with *check_http* plugin. PostgreSQL is monitored (only connection accessibility) using *check_postgres* plugin [17], which establishes connection to databasse and sending simple query (version check). Elasticsearch database can be monitored by its HTTP API documented here [18]. Plugins in Ruby language were developed to monitor this database. State of Logstash service is checked using NRPE and script placed locally on ELK host.

## 5. Conclusion

This article presents methodologies and technologies that might be used to integrate Nagios monitoring software with VMware virtualization stack. Monitoring solutions provided by VMware have certain disadvatanges - cost and limitation of scope of monitoring. Nagios eliminates those drawbacks, by allowing system administrator to integrate it into every layer of infrastructure (not only those based on VMware stack). Basic version of Nagios (named Nagios Core) can be used free of charge, thus eliminating cost disadvantage of vCOps. Monitoring system presented in this paper gives an overview of state of whole infrastructure and notifies system administrators about critical events. Methods presented, are de facto standards (SNMP protocol,

WMI, vSphere API, etc.) used to manage infrastructures and that gives provider a certain level of reliability and stability.

Further work can be done by improving monitoring capabilities of the system. Methodologies to integrate Nagios with other virtualization stacks and with specific server software (e.g. databases, HTTP server) can be researched. Also data produced by monitoring system (hardware and software metrics, performance data etc.) can be used by other researchers as the test data for machine learning experiments. In this kind of data certain patterns can be found, for example regular spikes in CPU load present regular backups of the infrastructure data. Another type of pattern can be missing metrics, which can represent some errors in working system. Patterns found in this data are not trivial, because of the fact that they can present different types of events in the infrastructure - deterministic (e.g. planned backup) or nondeterministic events (e.g. error in databases). Further research possibly can provide ways to automate diagnosis of those problems and determining their type.

## References

[1] P. Mell, T. Grance: *The NIST Definition of Cloud Computing*, NIST Special Publiction 800-145

[2] J. Spring: *Monitoring Cloud Computing by Layer, Part 1*, IEEE Security & Privacy March/April 2011

[3] J. Spring: *Monitoring Cloud Computing by Layer, Part 2*, IEEE Security & Privacy May/June 2011

[4] vCenter Operations Manager: [https://www.vmware.com/support/pubs/vcops-pubs.html]

[5] Nagios: [http://www.nagios.org/]

[6] Nagios Remote Plugin Executor: [http://exchange.nagios.org/directory/Addons/Monitoring-Agents/NRPE--2D-Nagios-Remote-Plugin-Executor/details]

[7] RFC 1448: *Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)* [https://tools.ietf.org/html/rfc1905]

[8] vSphere ESXi: [https://www.vmware.com/products/vsphere/features/esxi-hypervisor]

[9] vSphere 5.5 API: [https://pubs.vmware.com/vsphere-55/index.jsp]

[10] vSphere 5.5 API - Performance metrics: [https://www.vmware.com/support/developer/converter-sdk/conv55_apireference/vim.PerformanceManager.html]

[11] RbVmomi: [https://github.com/vmware/rbvmomi]

[12] Window Management Instrumentation: [https://msdn.microsoft.com/library/aa394582.aspx]
[13] wmic: [http://www.aldeid.com/wiki/Wmic-linux]
[14] Cacti: [http://www.cacti.net/]
[15] RRDtool: [http://oss.oetiker.ch/rrdtool/doc/index.en.html]
[16] Check WMI Plus: [http://exchange.nagios.org/directory/Plugins/Operating-Systems/Windows/WMI/Check-WMI-Plus/details]
[17] check_postgres: [https://github.com/bucardo/check_postgres]
[18] Elasticsearch API: [http://www.elastic.co/guide/]

# MONITOROWANIE INFRASTRUKTURY CLOUD COMPUTING OPARTEJ O NADZORCĘ VMWARE PRZY UŻYCIU NAGIOSA

**Streszczenie** Cloud computing (CC) jest w dzisiejszych czasach bardzo popularnym modelem usług. W tym modelu, klient posiada dostęp do dynamicznej platformy lub infrastruktury do wykonywania różnych zadań. Monitorowanie infrastruktury będącej podstawą takiej usługi, pozwala usługodawcom na zapewnienie wysokiej jakości usługi. Ten artykuł opisuje wymagania oraz metody implementacji systemu monitorowania, bazując na oprogramowaniu Nagios, dla infrastruktury CC opartej o oprogramowanie do wirtualizacji firmy VMware. Dodatkowo przedstawia on studium przypadku takiego systemu.

**Słowa kluczowe:** monitorowanie, VMware, Nagios, cloud computing