Network Planning and Management (PGRE)
Master in Electrical and Computers Engineering
Faculty of Engineering of University of Porto

Network and system management platforms

AFONSO QUEIRÓS UP201808903
HUGO GUIA UP201305075
JOÃO LOUREIRO UP201604453

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

28-03-2020

Contents

1 Introduction **1**

2 Services Configuration **2**
2.1 Apache2 - Web Service 2
2.2 VSFTPD - FTP Service 2
2.3 NTP - Network Time Protocol 4
2.4 PostFix - mail transfer agent 4
2.5 Bind9 DNS cache server 7

3 Nagios Configuration **8**

4 Zabbix Configuration **10**

5 Results **12**
5.1 Nagios Results 12
5.2 Zabbix Results 15
5.3 Theoretical Comparison: Grafana vs OpenDCIM 16
5.3.1 Grafana 16
5.3.2 OpenDCIM 17

6 Conclusion **19**

1. Introduction

The objective of this work is to study the potentialities in the use of freeware public tools for the management of equipment or services, and in particular in the monitoring component. The monitoring tools that we used are: Nagios, which is an older but reliable tool with great performance but is also very plug-in dependant, and Zabbix, which is a new tool but its community is gaining momentum mainly because of the UI being a lot more practical.

Using the available servers the following set of services that are normal to exist on a production network were set up: a Web server, a FTP server, a NTP server, an E-mail server and a DNS cache server.

In order to test these services and to make sure everything is running properly some failures and errors were forced on the servers' end.

2. Services Configuration

To perform the experiment, three servers are used, tux21, tux22 and tux23. Tux21 contains the DNS tool,tux22 contains FTP and SMTP service and tux23 presents Web service and NTP. Both Nagios and Zabbix allows configuration parameters that make periodic requests to server, in order to capture information about the services.

2.1 Apache2 - Web Service

For this service, the command "apt-get install apache2" was enough to do the job, since php and cgi dependencies are already downloaded. The next figure shows the current status of the service

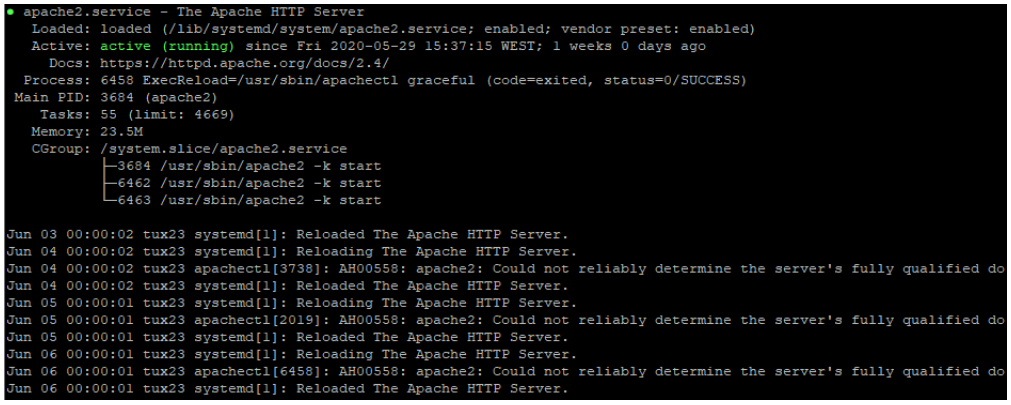

Figure 2.1: Current Status of Apache2 service.

With that said, the HTTP service will be ready to be used. Using the command 'service apache2 restart', the service will reboot with the desired settings. The following figure shows how the service works, for the conditions used.

2.2 VSFTPD - FTP Service

The second service used, VSFTPD, uses the FTP protocol (file transfer). It also needs to be configured to be used in accordance with our system. The settings added to the file located in /etc/vsftpd.conf, for the correct functionality of the system are the follows:

```
# Example config file /etc/vsftpd.conf
#
# The default compiled in settings are fairly paranoid. This sample file
# loosens things up a bit, to make the ftp daemon more usable.
# Please see vsftpd.conf.5 for all compiled in defaults.
#
# READ THIS: This example file is NOT an exhaustive list of vsftpd options.
# Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's
# capabilities.
#
#
# Run standalone?  vsftpd can run either from an inetd or as a standalone
# daemon started from an initscript.
listen=NO
#
# This directive enables listening on IPv6 sockets. By default, listening
# on the IPv6 "any" address (::) will accept connections from both IPv6
# and IPv4 clients. It is not necessary to listen on *both* IPv4 and IPv6
# sockets. If you want that (perhaps because you want to listen on specific
# addresses) then you must run two copies of vsftpd with two configuration
# files.
listen_ipv6=YES
#
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=YES
#
# Uncomment this to allow local users to log in.
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
write_enable=YES
#
# Default umask for local users is 077. You may wish to change this to 022,
# if your users expect that (022 is used by most other ftpd's)
#local_umask=022
#
# Uncomment this to allow the anonymous FTP user to upload files. This only
# has an effect if the above global write enable is activated. Also, you will
# obviously need to create a directory writable by the FTP user.
anon_upload_enable=YES
#
# Uncomment this if you want the anonymous FTP user to be able to create
# new directories.
anon_mkdir_write_enable=YES
#
# Activate directory messages - messages given to remote users when they
# go into a certain directory.
dirmessage_enable=YES
#
# If enabled, vsftpd will display directory listings with the time
# in  your  local  time  zone.  The default is to display GMT. The
# times returned by the MDTM FTP command are also affected by this
# option.
use_localtime=YES
#
# Activate logging of uploads/downloads.
xferlog_enable=YES
#
```

Figure 2.2: Configuration of VSFTPD.

Given this configuration, it is then possible to start using the service. The following figure shows the status of the service, emphasizing that it is active, allowing the user to transfer files using this tool.

```
root@tux22:~# service vsftpd status
● vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2020-06-05 11:56:30 WEST; 1 day 3h ago
  Process: 31077 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty (code=exited, status=0/SUCCESS)
 Main PID: 31078 (vsftpd)
    Tasks: 1 (limit: 4669)
   Memory: 2.5M
   CGroup: /system.slice/vsftpd.service
           └─31078 /usr/sbin/vsftpd /etc/vsftpd.conf

Jun 05 11:56:30 tux22 systemd[1]: Starting vsftpd FTP server...
Jun 05 11:56:30 tux22 systemd[1]: Started vsftpd FTP server.
```

Figure 2.3: Status of VSFTPD.

2.3 NTP - Network Time Protocol

The Network Time Protocol (NTP) is a networking protocol for clock synchronization between computer systems over packet-switched variable-latency data networks.

The configuration of NTP is located on etc/default/ntpdate. The next figure represents the configuration created.

```
# The settings in this file are used by the program ntpdate-debian, but not
# by the upstream program ntpdate.

# Set to "yes" to take the server list from /etc/ntp.conf, from package ntp,
# so you only have to keep it in one place.
NTPDATE_USE_NTP_CONF=yes

# List of NTP servers to use  (Separate multiple servers with spaces.)
# Not used if NTPDATE_USE_NTP_CONF is yes.
NTPSERVERS="0.debian.pool.ntp.org 1.debian.pool.ntp.org 2.debian.pool.ntp.org 3.debian.pool.ntp.org"

# Additional options to pass to ntpdate
NTPOPTIONS=""
```

Figure 2.4: Configuration of NTP.

For demonstrate the correct behaviour, the next figure represents the status of NTP service

```
● ntp.service - Network Time Service
   Loaded: loaded (/lib/systemd/system/ntp.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2020-05-29 15:52:45 WEST; 1 weeks 0 days ago
     Docs: man:ntpd(8)
 Main PID: 5036 (ntpd)
    Tasks: 2 (limit: 4669)
   Memory: 1.9M
   CGroup: /system.slice/ntp.service
           └─5036 /usr/sbin/ntpd -p /var/run/ntpd.pid -g -u 113:119

May 29 15:52:51 tux23 ntpd[5036]: Soliciting pool server 122.252.188.99
May 29 15:52:52 tux23 ntpd[5036]: Soliciting pool server 69.89.207.99
May 29 16:03:01 tux23 ntpd[5036]: 122.252.188.99 local addr 172.16.1.23 -> <null>
May 29 16:13:07 tux23 ntpd[5036]: 130.208.87.149 local addr 172.16.1.23 -> <null>
May 31 15:52:45 tux23 ntpd[5036]: leapsecond file ('/usr/share/zoneinfo/leap-seconds.list'): will expire in less than
Jun 01 15:52:45 tux23 ntpd[5036]: leapsecond file ('/usr/share/zoneinfo/leap-seconds.list'): will expire in less than
Jun 02 15:52:45 tux23 ntpd[5036]: leapsecond file ('/usr/share/zoneinfo/leap-seconds.list'): will expire in less than
Jun 03 15:52:45 tux23 ntpd[5036]: leapsecond file ('/usr/share/zoneinfo/leap-seconds.list'): will expire in less than
Jun 04 15:52:45 tux23 ntpd[5036]: leapsecond file ('/usr/share/zoneinfo/leap-seconds.list'): will expire in less than
Jun 05 15:52:45 tux23 ntpd[5036]: leapsecond file ('/usr/share/zoneinfo/leap-seconds.list'): will expire in less than
```

Figure 2.5: Status of NTP.

2.4 PostFix - mail transfer agent

As for the email service we used Postfix and had to configure it accordingly to our network topology. The main settings added to the file located at /etc/postfix/main.cf are the following:

```
# See /usr/share/postfix/main.cf.dist for a commented, more complete version


# Debian specific:  Specifying a file name will cause the first
# line of that file to be used as the name.  The Debian default
# is /etc/mailname.
#myorigin = /etc/mailname

smtpd_banner = $myhostname ESMTP $mail_name (Debian/GNU)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

readme_directory = no

# See http://www.postfix.org/COMPATIBILITY_README.html -- default to 2 on
# fresh installs.
compatibility_level = 2



# TLS parameters
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
smtpd_use_tls=yes
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache

# See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc package for
# information on enabling SSL in the smtp client.

smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated defer_unauth_destination
myhostname = tux22.netlab.fe.up.pt
mydomain = netlab.fe.up.pt
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = tux22.netlab.fe.up.pt, localhost.$mydomain, localhost, $mydomain
relayhost =
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128, 172.16.1.0/24
mynetworks_style = subnet
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
default_transport = error
relay_transport = error
inet_protocols = all
mail_spool_directory = /var/mail
```

Figure 2.6: Postfix configuration.

After editing the file, it is possible to test the service to see if the client is able to to successfully send an email. The service is restarted and the command " echo 'init' | mail -s 'init' root " is executed just to test

if the target is receiving the message properly.

This command sends a simple message written "init" in order to test the platform.

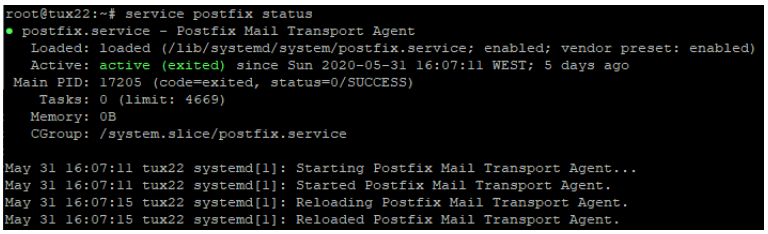

Figure 2.7: Status of Postfix service.

2.5 Bind9 DNS cache server

The Domain Name System (DNS) service required the BIND Domain Server of bind9utils which is running properly after being slightly modified.

The forwarder is now set to our Lab's DNS server IP address which is 172.16.1.1.

```
acl goodclients{
        localhost;
        localnets;
        172.16.1.0/24;
};

options {
        directory "/var/cache/bind";
        recursion yes;
        allow-query { goodclients; };
        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        forwarders {
                172.16.1.1;
                1.1.1.1;
                8.8.8.8;
        };

        forward only;

        //========================================================================
        // If BIND logs error messages about the root key being expired,
        // you will need to update your keys.  See https://www.isc.org/bind-keys
        //========================================================================
        dnssec-enable yes;
        dnssec-validation yes;

        auth-nxdomain no;    # conform to RFC1035
        listen-on-v6 { any; };
};
```

Figure 2.8: Bind9utils configuration file (bind9.options).

```
tux21:~# service bind9 status
● bind9.service - BIND Domain Name Server
   Loaded: loaded (/lib/systemd/system/bind9.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2020-05-31 15:30:15 WEST; 5 days ago
     Docs: man:named(8)
 Main PID: 21585 (named)
   CGroup: /system.slice/bind9.service
           └─21585 /usr/sbin/named -f -4 -u bind

May 31 15:30:19 tux21 named[21585]: GeoIP NetSpeed (type 10) DB not available
May 31 15:30:19 tux21 named[21585]: using default UDP/IPv4 port range: [32768, 61000]
May 31 15:30:19 tux21 rndc[21599]: server reload successful
May 31 15:30:19 tux21 systemd[1]: Reloaded BIND Domain Name Server.
May 31 15:30:19 tux21 named[21585]: sizing zone task pool based on 6 zones
May 31 15:30:19 tux21 named[21585]: configuring command channel from '/etc/bind/rndc.key'
May 31 15:30:19 tux21 named[21585]: reloading configuration succeeded
May 31 15:30:19 tux21 named[21585]: reloading zones succeeded
May 31 15:30:19 tux21 named[21585]: all zones loaded
May 31 15:30:19 tux21 named[21585]: running
```

Figure 2.9: Status of DNS service.

3. Nagios Configuration

In order to configure the Nagios Core tool, some dependencies were installed. Plug-ins embedded to the software, apache2, php are some of them. Next, NRPE, the engine that runs the plug-ins, is installed. These are the basis of how the program works. The server where the service is located has manual settings on the three clients, where it specifies the addresses of each machine. This software is widely used in the industry because:

- it uses secured monitoring, using SSH or SSL;
- Simple plugin development that allows users to easily create their own monitoring modes depending on their needs, using the development tool of their choice;
- Services are "checked-paralleled" - Even the system has 1000 services, there is no process lag;
- Ability to notify when a service or equipment has problems and when the problem is solved (via email, pager, SMS, or any other means defined by the user by plugin);
- user-friendly interface to present statistics about system architecture.

An example of configuring the client on the server is the image below.

The setup of Nagios included the following commands:

- apt install apache2
  wget https://github.com/NagiosEnterprises/nagioscore/archive/nagios-4.4.5.tar.gz
  tar xzf nagios-4.4.5.tar.gz
  cd nagioscore-nagios-4.4.5/"
  -> in order to install apache2, download Nagios and extract the downloaded content into a folder
- ./configure –with-httpd-conf=/etc/apache2/sites-enabled
  make all
  -> compile Nagios source code and define the Apache virtual host configuration for Nagios
- make install-groups-users
  usermod -a -G nagios www-data
  -> create the Nagios user and group, and add the 'www-data' Apache user to the 'nagios' group.
- make install
  make install-daemoninit
  make install-commandmode
  -> install Nagios binaries, service daemon script, and the command mode.
- make install-config
  ->install the sample script configuration.
- make install-webconf
  a2enmod rewrite cgi
  ->install the Apache configuration for Nagios and activate the mod_rewrite and mode_cgi modules.

Setting up on the client side involves the installation of Nagios NRPE server using "apt install nagios-nrpe-server nagios-plugins". After that, we can edit the file on "/etc/nagios/nrpe.cfg" and add the following information:

- server_address=172.16.1.21 -> in the case of tux21, ...1.22 for tux22, etc
- allowed_hosts=127.0.0.1,::1,172.16.1.24 -> to allow traffic from tux24

There is also the need to change "/etc/nagios/nrpe_local.cfg" to indicate the services that we want to be monitored periodically. That information can be found in the following picture.

```
##################################
# Do any local nrpe configuration here
##################################
command[check_root]=/usr/lib/nagios/plugins/check_disk -w 20% -c 10% -p /
command[check_ping]=/usr/lib/nagios/plugins/check_ping -H 172.16.1.21 -w 100.0,20% -c 500.0,60% -p 5
command[check_ssh]=/usr/lib/nagios/plugins/check_ssh -4 172.16.1.21
command[check_http]=/usr/lib/nagios/plugins/check_http -I 172.16.1.21
command[check_apt]=/usr/lib/nagios/plugins/check_apt
command[check_dns]=/usr/lib/nagios/plugins/check_dns -H 172.16.1.21
```

Figure 3.1: Configuration of Tux21 on the Client side.

On the server side, the file "/usr/local/nagios/etc/servers/client01.cfg" had to be created containing information related with the services on the client side. That information can be found in the following picture.

```
# Ubuntu Host configuration file1

define host {
        use                             linux-server
        host_name                       tux21
        alias                           tux21
        address                         172.16.1.21
        register                        1
}

define service {
      host_name                         tux21
      service_description               Check Ping
      check_command                     check_nrpe!check_ping
      max_check_attempts                2
      check_interval                    2
      retry_interval                    2
      check_period                      24x7
      check_freshness                   1
      contact_groups                    admins
      notification_interval             2
      notification_period               24x7
      notifications_enabled             1
      register                          1
}

define service {
      host_name                         tux21
      service_description               Check SSH
      check_command                     check_nrpe!check_ssh
      max_check_attempts                2
      check_interval                    2
      retry_interval                    2
      check_period                      24x7
      check_freshness                   1
      contact_groups                    admins
      notification_interval             2
      notification_period               24x7
      notifications_enabled             1
      register                          1
}

define service {
      host_name                         tux21
      service_description               Check DNS
      check_command                     check_nrpe!check_dns
      max_check_attempts                2
      check_interval                    2
      retry_interval                    2
      check_period                      24x7
      check_freshness                   1
      contact_groups                    admins
      notification_interval             2
      notification_period               24x7
      notifications_enabled             1
      register                          1
}
```

Figure 3.2: Configuration of Tux21 on the Server side.

4. Zabbix Configuration

Contrarly to the software we configured previously in this subject which only included some package installation and subsequent configuration, the setup of the Zabbix software required a MySQL database. So, we installed the software MariaDB on the server side, tux24, to handle the database management via "sudo apt-get install mariadb-server -y". After configuring a database with pre-defined credentials, we could then proceed to setup Zabbix. On the server side, we installed zabbix-server using the command: "apt install zabbix-server-mysql zabbix-frontend-php -y" and on the client side, ie. other tux computers, the zabbix-agent via "apt install zabbix-agent -y".

On the client side, the file located in "/etc/zabbix/zabbix_agentd.conf" needs to be updated with the following:

- Server=172.16.1.24 -> pointing to the server IP
- ServerActive=172.16.1.24 -> pointing to the active server IP (the same as before)
- Hostname=tux21 -> one different for each tux

On the server side, the file in "/etc/zabbix/zabbix_server.conf" needs to be updated with the database credentials we setup previously.

After this steps, we were presented with a GUI where we could setup extra details about the software such as its hostname and port.

Below is the Zabbix dashboard which shows all systems up and running.

Figure 4.1: Zabbix's dashboard with all hosts working

```
##### Passive checks related

### Option: Server
#       List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers and Zabbix proxies.
#       Incoming connections will be accepted only from the hosts listed here.
#       If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally
#       and '::/0' will allow any IPv4 or IPv6 address.
#       '0.0.0.0/0' can be used to allow any IPv4 address.
#       Example: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
#
# Mandatory: yes, if StartAgents is not explicitly set to 0
# Default:
# Server=

Server=172.16.1.24

### Option: ServerActive
#       List of comma delimited IP:port (or DNS name:port) pairs of Zabbix servers and Zabbix proxies for active checks.
#       If port is not specified, default port is used.
#       IPv6 addresses must be enclosed in square brackets if port for that host is specified.
#       If port is not specified, square brackets for IPv6 addresses are optional.
#       If this parameter is not specified, active checks are disabled.
#       Example: ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,::1,[12fc::1]
#
# Mandatory: no
# Default:
# ServerActive=

ServerActive=172.16.1.24

### Option: Hostname
#       Unique, case sensitive hostname.
#       Required for active checks and must match hostname as configured on the server.
#       Value is acquired from HostnameItem if undefined.
#
# Mandatory: no
# Default:
Hostname=tux23
```

Figure 4.2: Zabbix's configuration agent

5. Results

5.1 Nagios Results

In order to be able to demonstrate the correct functioning of the tool, some tests were carried out with all 3 clients, where in some cases all services of each machine are turned off and where only one of them is put on stand-by.

The first figure represents tux21 down with all services down, the next figure represents only one service down (in this case, FTP) in tux22. This results demonstrate how volatile can be this tool, responding well to changes in the machines, and can be very precise.

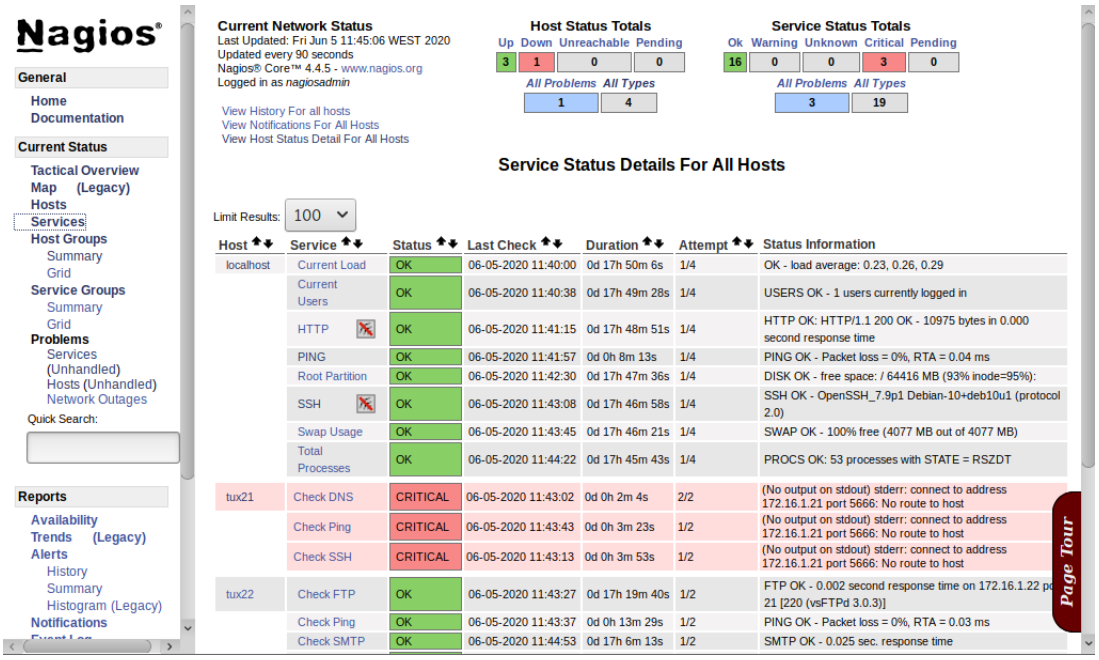

Figure 5.1: TUX21 down with all services down.

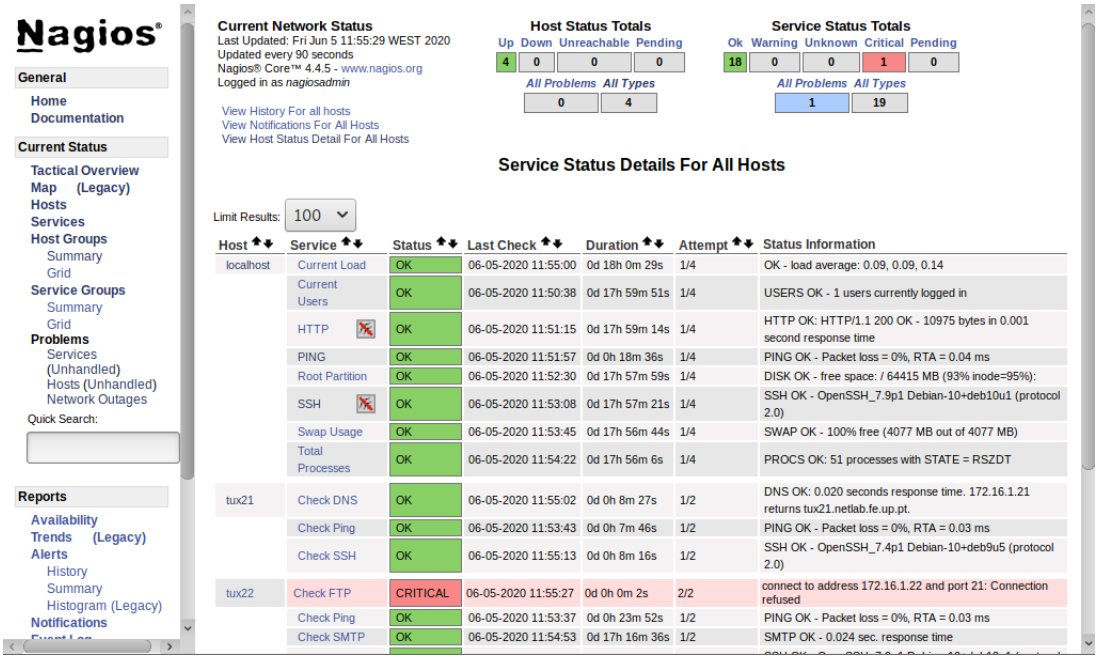

Figure 5.2: TUX22 down with FTP service down.

This tool has another point of view of the architecture, more precisely graphics showing status of the machines communicating with the server.This way, user can have a more interactive interface, checking if all machines are communicating successfully with the server.

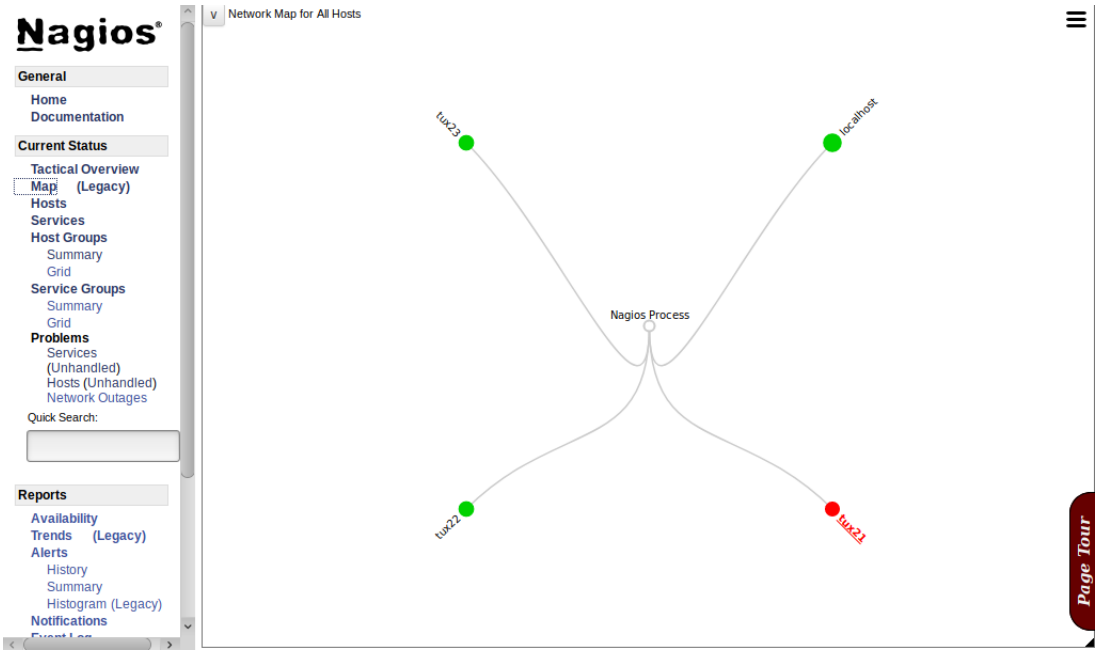

Figure 5.3: View of Network Architecture with TUX21 down .

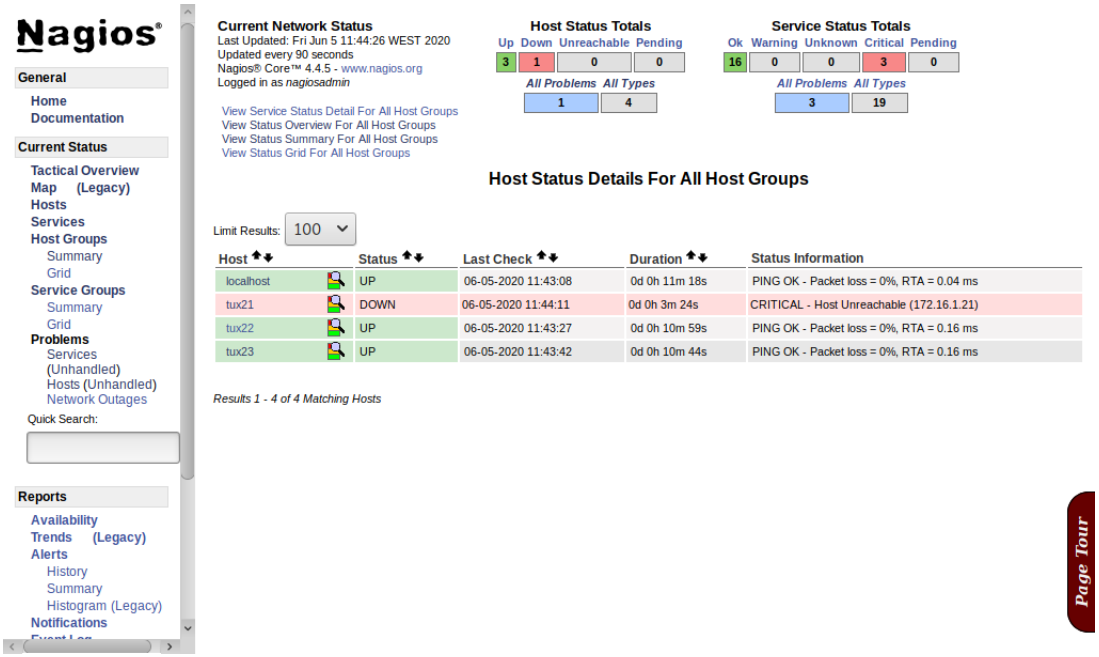

Figure 5.4: Resume of Architecture Statistics.

With the results presented, we can say that this tool is very responsive to network failures or changes to it, executing tasks in a short period of time, in order to inform the user of the system's behavior in time. Due to its simplicity of presenting values, it is very useful and necessary when it comes to service management.

5.2 Zabbix Results

In order to be able to demonstrate the correct functioning of the tool, some tests were carried out with all 3 clients, just like it was done for Nagios.

The first figure represents tux21 down with all services down, the next figure represents only one service down (in this case, FTP) in tux22, and the third shows the service ZBX down (tux21). Snaps of the Zabbix dashboard were taken in order to show the simplicity and practicality of this tool.

Figure 5.5: Zabbix's dashboard with tux22 down (all services)

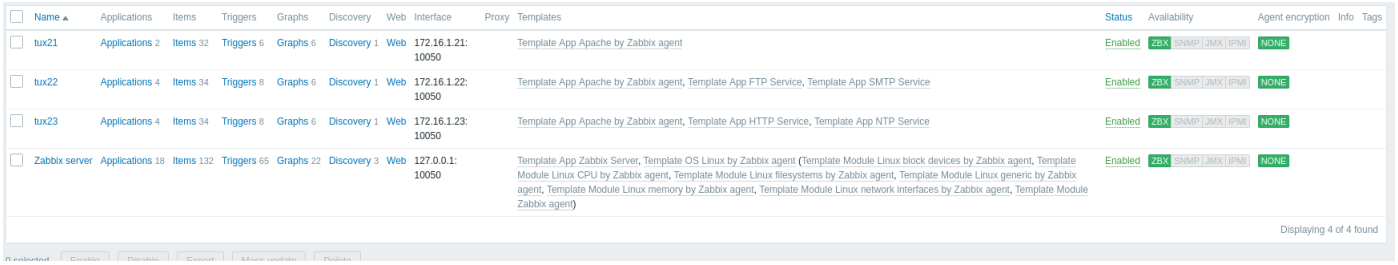

Figure 5.6: All configured hosts up

In the image below we can see FTP has stopped working on tux22 and appears under the "problems" tab right under the Global View UI.

Figure 5.7: Zabbix's dashboard with tux21's FTP service down

ZBX is the control protocol in charge of maintaining the connection between the administrator operating the tool and tux21 itself. In this capture it shows that we lost connection to tux21 at that moment.

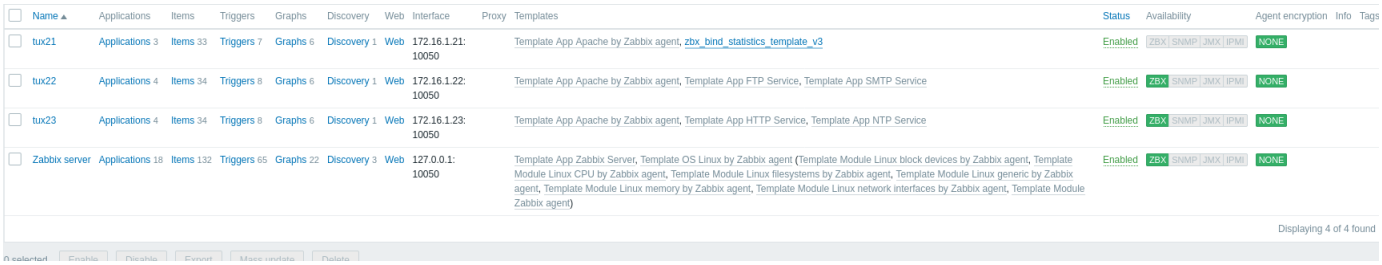

Figure 5.8: Zabbix's dashboard with ZBX down on tux21

5.3 Theoretical Comparison: Grafana vs OpenDCIM

In addition to these two tools presented, there are other ways to assess and monitor the status of a network. Grafana and OpenDCIM are two concrete examples. In this section, each tool will be briefly presented, highlighting the advantages and disadvantages of each one.

5.3.1 Grafana

Grafana is a platform for visualizing and analyzing metrics using graphs. It supports several types of databases - both free and paid - and can be installed on any operating system. To facilitate the visualization of the graphs, it is possible to create dynamic dashboards. Supports multiple user viewing. In addition, the tool allows you to configure alerts based on the metrics, which are continuously analyzed to notify the user whenever necessary, according to the rules defined by him. It is widely used by monitoring systems to generate real-time graphics. It has a user-friendly interface, for better understanding and response to drastic changes. Unique features:

- Light system structure;
- It allows organizing data from different sources in the same dashboard.
- All plugins can be installed and managed simply in the tool, and there are many options available.
- **Rich and participatory community:** open source system. There are quite a few responsible for making improvements to the system, and even creating different and personalized interfaces, for different tastes and needs.
- **Integration with other tools:** possibility to integrate the system with other data tools, such as MySQL or PostgreSQL.

Figure 5.9: Grafana Interface.

As we can see, this tool is complete, showing a lot of statistical results in just one dashboard, through graphics, for easier get of information. Comparated with Nagios and Zabbix, is a competitive solution, and the instalation is also easy to make, because the software has a lot of tutorials and support information in internet.

5.3.2 OpenDCIM

OpenDCIM is a more basic solution, in comparasion with the other 3 tools, but is a great tool for monitoring too. It is an open source Data Center Inventory Management (DCIM) application. Unique features:

- Image mapping with custom image for creating click-able zones for each cabinet;
- Overlay layers on map for Power, Space, Temperature, and Weight capacity;
- Multiple levels of user rights;
- Support for automatic transfer switches.

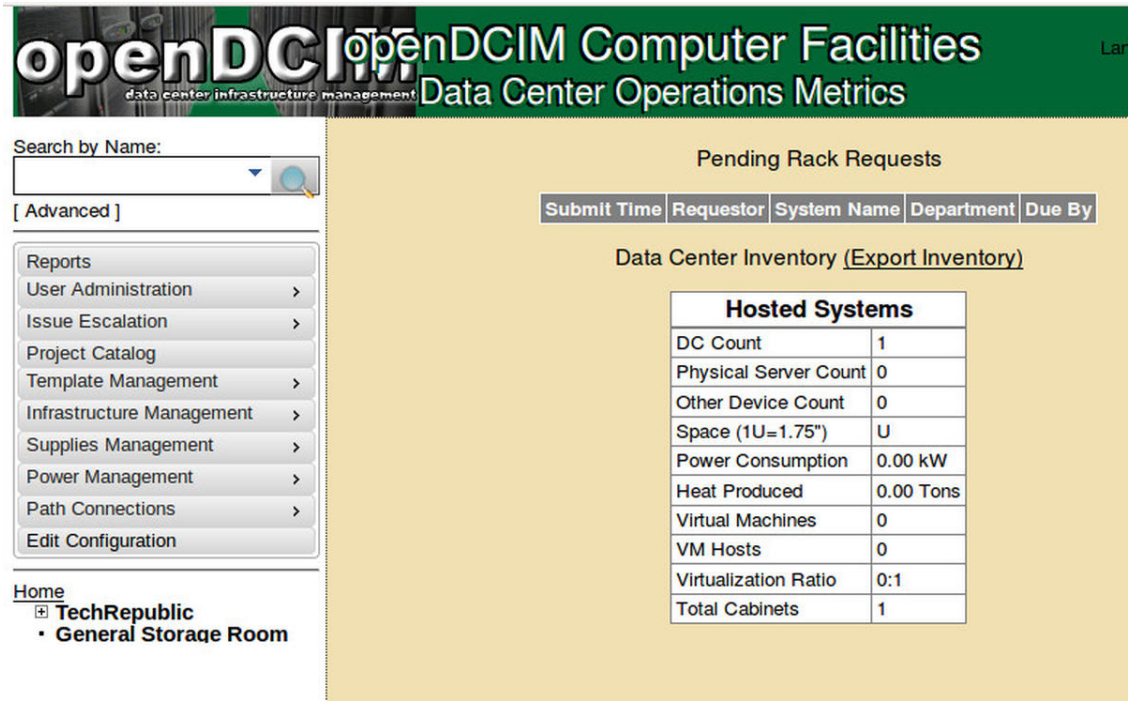

Figure 5.10: OpenDCIM Interface.

As we can see, this tool is orientated to process other information, such like temperature of power of one machine. This tool is not oriented to process information about services running on it, but it can be edited to perform such tasks. It has a similar installation, in terms of complexity, related with other 3 tools. In conclusion, this tool is not oriented to monitoring the same information as 3 tools mentioned before, but it offers another type of experience to the user, allowing to monitoring physical variables in a network system, like racks space and placement, or switch and router statistics.

6. Conclusion

After we got familiar with both platforms we can conclude that both are great monitoring tools even though Zabbix takes the upper hand here. Both Nagios and Zabbix are great networking tools which focus more on monitoring and statistical analysis, but there's some aspects that we cant overlook, for instance, Nagios requires a lot of plugins in order to achieve its maximum potential, whereas Zabbix comes with all functionalities already included as well as it is much easier to use and analyze. The UI and graph analysis are definitely much better in Zabbix, however the fact that it does not have plugins can also be counted as a disadvantage.

As for the configuration aspect, Nagios behaved better against all odds in the sense that we had no problems with it, while Zabbix took longer then we expected. Zabbix also has better alerts and notifications with more options and configurations while Nagios is less flexible on that subject.

The protocol support is the same on both platforms (both include HTTP, FTP, SMTP, SNMP, POP3, SSH, and MySQL) altough Zabbix has multiple monitoring templates for FTP, HTTP, HTTPS, IMAP, LDAP, MySQL, NNTP, SMTP, SSH, POP and Telnet which helps reducing the configuration needs.

To conclude, Zabbix is the better tool for monitoring because it has a better UI and dashboard which makes monitoring itself a lot easier, as well as better graphical information for deeper analysis.