
Trabalho Laboratorial 5

Planeamento e Gestão de Redes

Plataformas de gestão de redes e sistemas

Diogo Remião & Miguel Pinheiro

Junho 2021



FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Faculdade de Engenharia da Universidade do Porto
TEC

Conteúdos

1	Introdução	2
2	Setup dos Serviços	3
2.1	HTTP Server	3
2.2	DNS Server	4
2.3	NTP Server	4
2.4	FTP Server	6
2.5	SNMP	6
2.6	Email Server	7
3	Nagios	8
3.1	Instalação e configuração	9
3.2	Resultados	12
3.3	Teste de falhas	13
4	Zabbix	15
4.1	Instalação e configuração	16
4.2	Resultados	20
4.3	Teste de falhas	21
5	Outras ferramentas	23
5.1	Grafana	23
5.2	openDCIM	24
6	Análise comparativa	25
7	Conclusão	27
	Bibliografia	28

1 Introdução

O objetivo de trabalho prende-se com a análise de **ferramentas de gestão e monitorização** de serviços de uma rede. Ao contrário de ferramentas como MRTG e NTOP que monitorizam o tráfego, neste trabalho serão abordadas ferramentas que monitorizam diretamente os diferentes sistemas e os serviços neles alojados.

As ferramentas utilizadas são o **Nagios Core** e **Zabbix**, ambas grátis e open-source. Será igualmente realizada uma análise às diferentes funcionalidades e capacidade de personalização de ambas as ferramentas num ambiente de teste criada na nossa bancada, onde serão alojados vários serviços nos diferentes computadores. Testes de falha de sistemas e serviços serão efetuados de modo a analisar o funcionamento das ferramentas na deteção de falhas.

Por fim, será feita uma análise comparativa entre estas ferramentas com duas outras alternativas no mercado.

2 Setup dos Serviços

Nesta secção são abordadas a alocação e configuração dos diferentes serviços nos diferentes computadores.

A tipologia de rede usada foi a seguinte:

Serviço	Computador
HTTP Server	Tux14
DNS Server	Tux14
FTP Server	Tux12
NTP Server	Tux12
Email Server	Tux12 / Tux14
SNMP	Router / Switch
Nagios	Tux13
Zabbix	Tux13

Table 2.1: Alocação dos serviços nos computadores

Todos os serviços instalados serão mencionados e testados, no entanto as suas instalações não serão abordadas dado que são iguais aos trabalhos anteriores.

2.1 HTTP Server

O servidor HTTP foi configurado com recurso ao **Apache** [1]. Este servidor corre uma página dinâmica PHP para impedir *caching* do seu conteúdo.

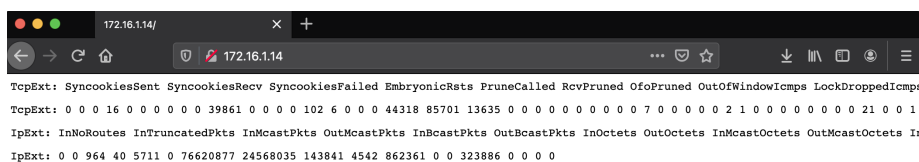


Figure 2.1: Servidor Apache no Tux14

2.2 DNS Server

O servidor DNS foi configurado com recurso ao **Bind** [2]. No servidor DNS foi configurado como **Cache DNS**, fazendo *forward* para endereços que não consegue resolver para os servidores DNS da *Google*. Na base de dados local foi adicionada a entrada do domínio `www.example.com`, com o respetivo IP `172.16.1.15`.

```
root@tux12:~# dig www.example.com

;<<>> DiG 9.11.5-P4-5.1+deb10u5-Debian <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2744
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: udp: 4096
; COOKIE: 84c58d584b9e518d28d6b843e0b21639cbdc52bb7699f6d8 (good)
;; QUESTION SECTION:
;www.example.com.                IN      A

;; ANSWER SECTION:
www.example.com.                604800  IN      A      172.16.1.15

;; AUTHORITY SECTION:
example.com.                    604800  IN      NS      localhost.

;; ADDITIONAL SECTION:
localhost.                      604800  IN      A      127.0.0.1
localhost.                      604800  IN      AAAA   ::1

;; Query time: 0 msec
;; SERVER: 172.16.1.14#53(172.16.1.14)
;; WHEN: Sat May 29 11:23:53 WEST 2021
;; MSG SIZE rcvd: 155

root@tux12:~#
```

(a) Resolução do endereço `www.example.com`

```
root@tux12:~# dig google.com

;<<>> DiG 9.11.5-P4-5.1+deb10u5-Debian <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17823
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 13, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: udp: 4096
; COOKIE: c896060a53abd1b6997808ad60b2161afc5386eed6900944 (good)
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.                    6       IN      A      142.250.185.14

;; AUTHORITY SECTION:
.                               65772   IN      NS      i.root-servers.net.
.                               65772   IN      NS      m.root-servers.net.
.                               65772   IN      NS      h.root-servers.net.
.                               65772   IN      NS      k.root-servers.net.
.                               65772   IN      NS      j.root-servers.net.
.                               65772   IN      NS      a.root-servers.net.
.                               65772   IN      NS      d.root-servers.net.
.                               65772   IN      NS      e.root-servers.net.
.                               65772   IN      NS      l.root-servers.net.
.                               65772   IN      NS      b.root-servers.net.
.                               65772   IN      NS      c.root-servers.net.
.                               65772   IN      NS      f.root-servers.net.
.                               65772   IN      NS      g.root-servers.net.

;; Query time: 0 msec
;; SERVER: 172.16.1.14#53(172.16.1.14)
;; WHEN: Sat May 29 11:23:22 WEST 2021
;; MSG SIZE rcvd: 294

root@tux12:~#
```

(b) Caching da queries

Figure 2.2

2.3 NTP Server

O servidor NTP foi configurado com recurso ao *daemon* NTP [3]. Para sincronização, foram definidos os 3 servidores NTP indicados para Portugal <https://support.ntp.org/bin/view/Servers/NTPPoolServers>. O servidor está alojado no *tux12*, mas um cliente foi criado no *tux13* para efeitos de teste.

```
root@tux12:~# ntpq -p
      remote           refid      st t when poll reach   delay   offset  jitter
=====
+195.22.17.7 (ft 178.33.203.107  3 u  960 1024 377    6.672   -3.097   1.688
+ntp1.flashdance 192.36.143.151  2 u  189 1024 377   86.539    9.167   4.410
*vmd46520.contab 79.133.44.131   2 u    7 1024 377   56.659    2.864   1.201
root@tux12:~#
```

(a) Resolução do endereço www.example.com

```
root@tux13:~# ntpq -p
      remote           refid      st t when poll reach   delay   offset  jitter
=====
*ntp_server       144.91.116.85   3 u  573 1024 377    0.169    2.940   2.199
root@tux13:~#
```

(b) Caching da queries

Figure 2.3

2.4 FTP Server

O servidor FTP foi configurado com recurso ao **vsftpd** [4]. Foram definidos os dados de autenticação e colocado um ficheiro txt para efeitos de teste.

```
[root@tux12:~# ftp -inv 172.16.1.12
Connected to 172.16.1.12.
220 Welcome to Group3 FTP service.
[ftp> user ftp1
331 Please specify the password.
[Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
[ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r-- 1 0 0 3785 May 26 18:01 testfile
226 Directory send OK.
[ftp> get testfile
local: testfile remote: testfile
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for testfile (3785 bytes).
226 Transfer complete.
3785 bytes received in 0.00 secs (48.1288 MB/s)
[ftp> exit
221 Goodbye.
root@tux12:~#
```

Figure 2.4: Acesso ao servidor FTP

2.5 SNMP

De modo a monitorizar o estado do **Switch** e do **Router**, foi ativado um agente SNMP nestes dispositivos [5]. Este agente permite que, com recurso a este protocolo, uma ferramenta de monitorização tenha acesso a um conjunto de estatísticas que definem o estado do dispositivo.

Para ativar o SNMP, foi executado o comando `snmp-server community public ro` com permissões apenas de leitura das variáveis.

2.6 Email Server

Dois servidores email forma configurados com recurso ao **Postfix** [6]. Foram configurados dois servidores para permitir testar o serviço enviando emails de um servidor para o outro.

```
root@tux12:~# mail -s test mail netedu@tux14.netlab.fe.up.pt
Cc:
Test Mail
root@tux12:~# █
```

(a) Envio de email do Tux12 para Tux14

```
From root@tux12 Sat May 29 11:56:45 2021
Return-Path: <root@tux12>
X-Original-To: netedu@tux14.netlab.fe.up.pt
Delivered-To: netedu@tux14.netlab.fe.up.pt
Received: from tux12.netlab.fe.up.pt (tux12 [172.16.1.12])
        by tux14.netlab.fe.up.pt (Postfix) with ESMTP id 068D3121439
        for <netedu@tux14.netlab.fe.up.pt>; Sat, 29 May 2021 11:56:45 +0100 (WEST)
Received: by tux12.netlab.fe.up.pt (Postfix, from userid 0)
        id E313F12148F; Sat, 29 May 2021 11:56:44 +0100 (WEST)
Subject: test
To: <mail@tux12>, <netedu@tux14.netlab.fe.up.pt>
X-Mailer: mail (GNU Mailutils 3.5)
Message-Id: <20210529105644.E313F12148F@tux12.netlab.fe.up.pt>
Date: Sat, 29 May 2021 11:56:44 +0100 (WEST)
From: root <root@tux12>

Test Mail
```

(b) Receção do email no Tux14

Figure 2.5

3 Nagios

Nagios Core é uma ferramenta de monitorização de sistemas grátis e *open-source* [7]. É também oferecido um serviço pago Nagios XI, que é construído sobre o sistema *core*.

Esta ferramenta permite a monitorização de vários serviços, atuando como um *scheduler* que executa periodicamente testes para verificar o estado dos serviços e sistemas. Estes testes são os **plugins**, *scripts* maioritariamente Perl, executáveis, desenvolvidos quer internamente, quer pela comunidade. Existem plugins para testar várias funcionalidades, desde o estado de um servidor HTTP à carga de utilização do CPU num servidor.

É disponibilizada uma interface Web, com recurso ao Apache, onde várias estatísticas são apresentadas para o utilizador, assim como alertas sobre sistemas que estejam *down*. É de notar que várias versões do *frontend* são disponibilizadas, aumentando a capacidade de customização do sistema.

Todas as configurações são feitas através de ficheiros *txt* no host do Nagios, pelo que não é possível configurar a ferramenta na sua interface Web.

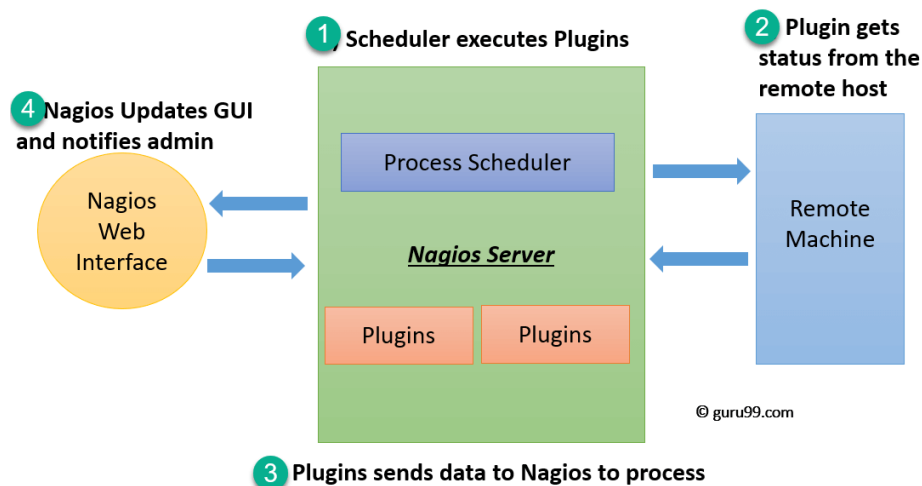


Figure 3.1: Princípio de funcionamento do Nagios

3.1 Instalação e configuração

A instalação foi feita segundo o guia de instalação do próprio Nagios [8] no Tux13. Foram definidos os dados de autenticação com o utilizador default *nagiosadmin*.

Após a configuração inicial, foi feita a configuração dos sistemas.

O primeiro passo é importar os **Plugins** do Nagios. Os seguintes *packages* foram instalados no servidor:

- **nagios-plugins**: os testes executáveis locais, i.e., que testam os serviços acedendo remotamente, por exemplo, a um servidor HTTP para verificar se está disponível.
- **nagios-nrpe-plugins**: os testes executáveis dos agentes, i.e., que são executados no sistema de destino para ter acesso a informação local, por exemplo, da utilização do CPU.

Devido aos objetivos deste trabalho consistirem maioritariamente com o teste dos serviços, foi utilizada a abordagem mais simples dos **nagios-plugins**, que não requer a instalação de agentes no computadores, apenas no servidor.

O segundo passo consiste em indicar ao Nagios quais os plugins utilizar e em que sistemas. Para efeitos de simplificação, foi criada a pasta `servers` no diretório `/usr/local/nagios/etc/servers`. Nesta pasta, foi criado um ficheiro `.cfg` para cada sistema, neste caso, `tux12.cfg`, `localhost.cfg`, `tux13.c`.

Nestes ficheiros é especificado quais são os plugins a executar em cada um dos computadores. Por exemplo, o ficheiro `tux12.cfg` ficou configurado da seguinte forma:

```
define host {
    use                linux-server
    host_name          tux12
    alias              FTP-NTP-Mail
    address            172.16.1.12
    register           1
}

define service {
    use                generic-service
    host_name          tux12
    service_description Mail
    check_command       check_smtp
    notifications_enabled 1
}

define service {
    use                generic-service
    host_name          tux12
    service_description Ping
    check_command       check_ping
    notifications_enabled 1
}

define service {
    use                generic-service
    host_name          tux12
    service_description FTP
    check_command       check_ftp
    notifications_enabled 1
}

define service {
    use                generic-service
    host_name          tux12
    service_description NTP
    check_command       check_ntp_time
    notifications_enabled 1
}

define service {
    use                generic-service
    host_name          tux12
    service_description SSH
    check_command       check_ssh
}
```

Note-se que primeiro é feita uma definição do host e do respetivo IP. Nos serviços, são identificados os testes a ser executados, especificando o host. É possível definir todos os testes e hosts no mesmo ficheiro, mas tal não é boa prática pois torna-se muito difícil modificar as configurações do sistema.

Dependendo do host e os serviços nele alojados, diferentes testes foram configurados:

Sistema	Testes
Tux12	check_ping check_ssh check_ntp check_ftp check_smtp (Mail)
Tux14	check_ping check_ssh check_http check_dns check_smtp (Mail)
Router	check_ping check_snmp_uptime_v2
Switch	check_ping check_snmp_uptime_v2
Tux13	localhost default tests

Table 3.1: Alocação dos serviços nos computadores

Dois testes requereram mais atenção.

Check_dns, apesar de ser um plugin instalado no *package*, não está configurado no ficheiro `commands.cfg`. Este ficheiro é onde se define a syntax para correr os testes. Muitos já estão configurados, mas este não. Desse modo configurou-se do seguinte modo:

```
define command {
    command_name    check_dns
    command_line    $USER1$/check_dns -H $ARG1$ -s $HOSTADDRESS$ -a $ARG2$
}
```

com `-H` o endereço a fazer a *query*, `-s` o servidor DNS a usar e `-a` o endereço IP esperado. Os argumentos são passados quando se define o teste nos ficheiros de configuração.

O teste predefinido do snmp, **check_snmp**, não funcionou, pelo que instalou manualmente um novo script **check_uptime**¹ para o mesmo efeito. Este script Perl foi importado para a pasta `\usr\lib\nagios\plugins`, sendo posteriormente definido como um executável para funcionar corretamente. Este foi configurado do seguinte modo no ficheiro `commands.cfg`:

```
define command {
    command_name check_snmp_uptime_v2
    command_line $USER1$/check_uptime.pl -2 -f -w -H $HOSTADDRESS$ -C public -T <-
    unix-sys
}
```

Este comando retorna o OID **sysUpTime** do SNMP no sistema, verificando assim o seu funcionamento.

3.2 Resultados

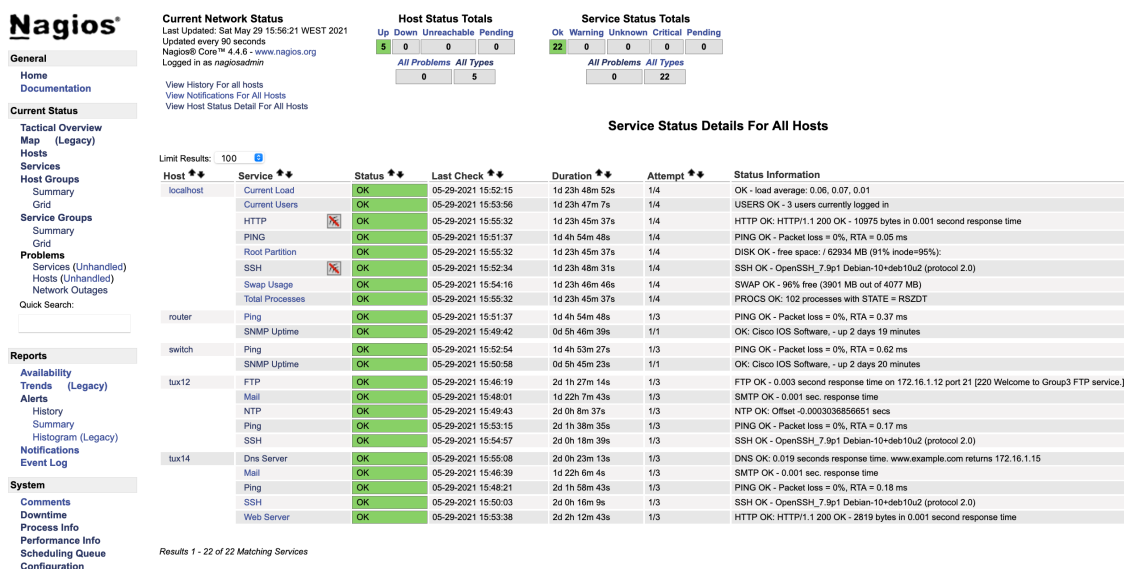


Figure 3.2: Interface Web com o status dos serviços e hosts

Como observado, todos os serviços estão **OK**. Também é apresentada informação relativamente ao *uptime* dos serviços, assim como o retorno dos testes.

O campo **Last Check** indica quando foi executado o último teste. É possível configurar o Nagios para aumentar a frequência de testes, no entanto isso aumenta também o tráfego interno de controlo pelo que é um *trade-off* a ter em conta.

Clicando em cada serviço ou host, é possível obter informação mais específica. No entanto, é uma *frontend* bastante simples e intuitiva.

¹https://exchange.nagios.org/directory/Plugins/System-Metrics/Uptime/check_uptime--2F-check_snmp_uptime/details

3.3 Teste de falhas

Primeiramente, simulou-se falhas nos serviços, parando alguns processos nos Tuxs:

Tux12

```
systemctl stop vsftpd - Falha do servidor FTP
systemctl stop postfix - Falha do servidor Email
```

Tux14

```
systemctl stop bind9 - Falha do servidor DNS
systemctl stop apache2 - Falha do servidor HTTP
```

Após algum tempo de atualização de informação, o output da interface do Nagios foi o seguinte:

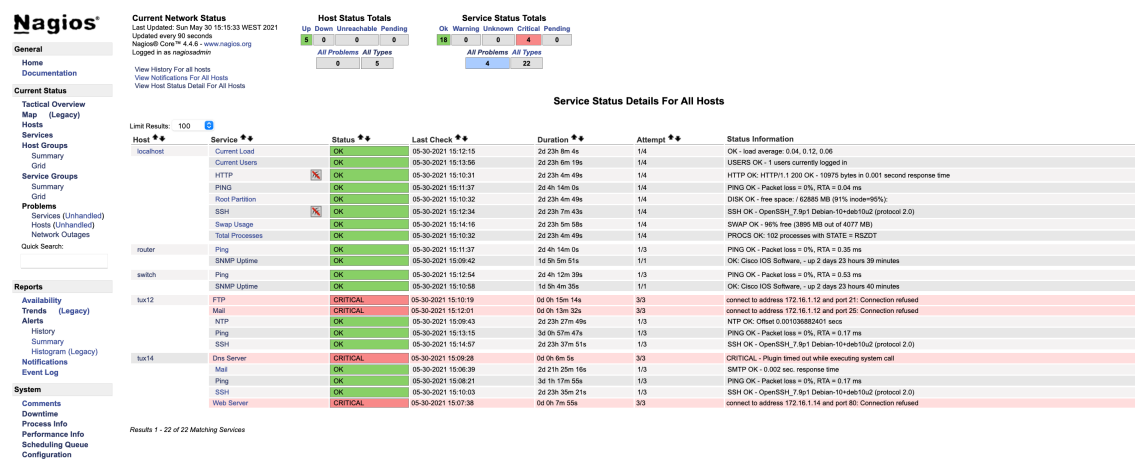


Figure 3.3: Falha de serviços

Como se observa, os serviços desativados estão **CRITICAL**, sendo possível observar o output dos plugins. É também apresentado quando tempo já passou desde que o serviço falhou.

De seguida simulou-se uma falha no Tux14. Para se simular este evento, configurou-se dois *cronjobs* consecutivos:

- ifconfig eth0 down: cortar a ligação do Tux14 à rede
- ifconfig eth0 up: ativar novamente a interface para retomar o acesso 15 minutos depois do anterior

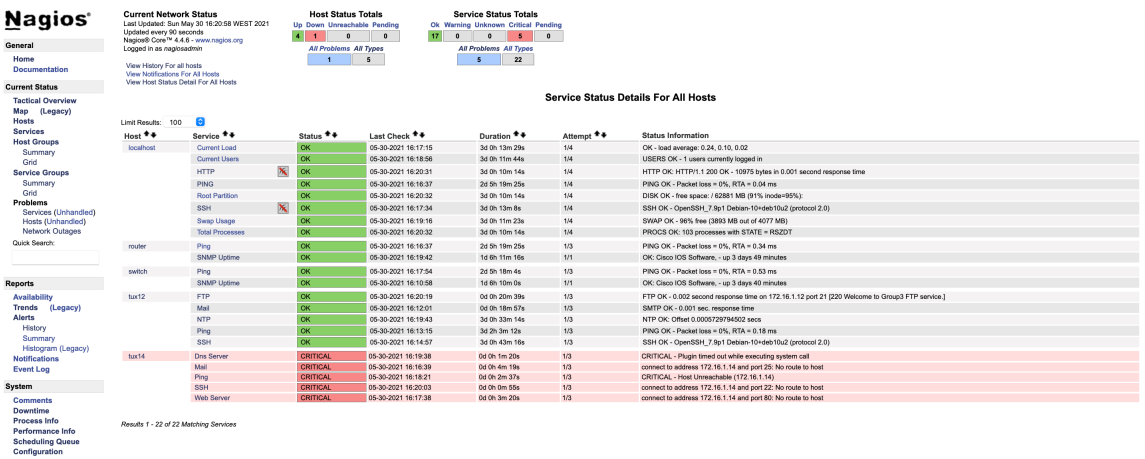


Figure 3.4: Falha do Tux14

Neste cenário, não só estão os serviços no estado **CRITICAL**, mas também os testes de conectividade Ping e SSH. O próprio host aparece a vermelho, e no topo da página é possível observar que há um host que está *down*.

O Nagios demorou cerca de 1 minuto a detetar que host estava down, que está relacionado com a frequência do teste configurada.

4 Zabbix

Zabbix, tal como o Nagios Core, é uma ferramenta de monitorização de sistemas de rede e informação *open-source* [9]. Permite a supervisão de diversos sistemas tais como redes, servidores VMs e serviços de cloud. Paralelamente a estes serviços, é também possível monitorizar o estado hardware em sistemas onde o Zabbix esteja instalado. O Zabbix funciona com apoio de uma base de dados. Se por predefinição, o Nagios usa MySQL, o Zabbix permite escolher de uma gama de opções, nomeadamente MySQL, MariaDB, PostgreSQL, etc.

O Zabbix não funciona à base de plugins, mas sim de **items**. Este items têm o mesmo princípio de funcionamento, consistindo em testes executados periodicamente para monitorizar uma característica específica. Os items podem ter outputs diferentes, quer seja números, strings ou booleanos.

No entanto, é de notar que o Zabbix não sabe interpretar o output destes items como correto ou não. Para se definir o estado de um serviço ou *feature*, é preciso definir os **triggers**.

Triggers correspondem a testes que são feitos ao output dos items. Por exemplo, se um determinado item tem como output possível **1/0**, um trigger pode definir 1 como OK e 0 como ERROR, dando desse modo feedback ao utilizar sobre o estado de serviço de uma forma mais direta.

Para simplificar o processo de setup, e evitar a configuração manual de todos os items e correspondentes triggers, são fornecidos **Templates** que contêm um conjunto de items e triggers adequados para determinados contextos. Por exemplo, o template **Template App HTTP Service** contém um item que testa um servidor HTTP, e um trigger que processa o output para determinar o seu estado. Alguns templates também contêm **dashboards**, que são uma compilação visual de informação relevante naquele template.

4.1 Instalação e configuração

A instalação foi feita de acordo com a documentação do Zabbix [10] no Tux13.

Similarmente ao Nagios, o Zabbix oferece duas opções de monitorização:

- **Zabbix-agent:** O Zabbix-agent é instalado no host destino, onde este coleta informação relevante do funcionamento do sistema, como a carga de CPU, disco, etc. O servidor Zabbix recebe depois esta informação diretamente do host, podendo assim mostrar várias estatísticas locais. É adequado quando queremos monitorizar um servidor ou computador.
- **Agentless:** Os testes são feitos apenas sobre serviços que possam ser acedidos externamente, por exemplo, um servidor HTTP. Não é preciso instalar nada, sendo que a monitorização destes sistemas recorre a protocolos de comunicação como HTTP, SSH, SNMP, TELNET, etc. É adequado quando queremos monitorizar um componente de *networking*, como um Router ou Switch.

Tendo isto consideração, a abordagem **Zabbix-agent** foi utilizada nos casos dos Tuxs, e a abordagem **Agentless** no Switch e no Router.

O próximo passo consiste na instalação dos diferentes *packages* requeridos pelo Zabbix. É aqui que se define também qual é a base de dados que se vai usar. A escolhida foi a **PostgreSQL**.

Foram definidos *Users* e as respetivas *passwords* quer para o Zabbix, quer para a base de dados. O passo seguinte consiste na criação da base de dados que contém todas as configurações do Zabbix ¹.

Após a configuração inicial, é definido no ficheiro `/etc/zabbix/zabbix_server.conf` os parâmetros da base de dados criada:

```
DBHost= (string nula para o caso do PostgreSQL)
DBName=zabbix
DBUser=zabbix
DBPassword=12345
```

Posteriormente, é feita a configuração final na página Web do Zabbix. Após este passo, é possível iniciar a configuração da monitorização da rede.

A instalação dos agentes no Tux12, Tux13 e Tux14 é feita com a instalação do dev-ido *package*. Posteriormente, o ficheiro `/etc/zabbix/zabbix_agentd.conf` é modificado, especificando-se nas linhas `"Server="` and `"ServerActive="` o IP do servidor Zabbix. O agente no Tux13 também é instalado pois é necessário para se poder fazer a monitorização do hardware do sistema no servidor.

¹A documentação online está errada a vários níveis, quer nos diretórios, quer na estrutura da base de dados. No entanto a documentação fornecida localmente após a instalação do respetivo *package* está correta

Ao contrário do Nagios, o Zabbix não é configurado localmente no sistema editando ficheiros de configuração, mas sim na interface Web do software. Logo à partida, é possível afirmar que, para o utilizador comum, é mais fácil usar uma UI do que recorrer extensivamente ao terminal.

Os **Hosts** são a primeira coisa a definir. A definição de um host consiste na definição da interface de comunicação entre o host e o servidor. Além do IP, é preciso definir se esses hosts tem ou não agente. No caso do Tux12 e Tux14, foi definida uma interface com agente. No caso do Switch e do Router, foi definida uma interface com recurso ao protocolo SNMP.

The screenshot shows the Zabbix Host configuration page for a host named 'tux12'. The 'Host name' field is set to 'tux12' and the 'Visible name' is also 'tux12'. The 'Groups' dropdown is set to 'Discovered hosts'. The 'Interfaces' table shows one interface with Type 'Agent', IP address '172.16.1.12', and Port '10050'. The 'Connect to' dropdown is set to 'IP'. The 'Default' checkbox is checked. The 'Description' field is empty. The 'Monitored by proxy' dropdown is set to '(no proxy)'. The 'Enabled' checkbox is checked. At the bottom, there are buttons for 'Update', 'Clone', 'Full clone', 'Delete', and 'Cancel'.

(a) Interface do Tux12 - Agent

The screenshot shows the Zabbix Host configuration page for a host named 'router'. The 'Host name' field is set to 'router' and the 'Visible name' is also 'router'. The 'Groups' dropdown is set to 'Templates/Network devices'. The 'Interfaces' table shows one interface with Type 'SNMP', IP address '172.16.1.19', and Port '161'. The 'Connect to' dropdown is set to 'IP'. The 'Default' checkbox is checked. The 'Description' field is empty. The 'Monitored by proxy' dropdown is set to '(no proxy)'. The 'Enabled' checkbox is checked. At the bottom, there are buttons for 'Update', 'Clone', 'Full clone', 'Delete', and 'Cancel'.

(b) Interface do Router - SNMP

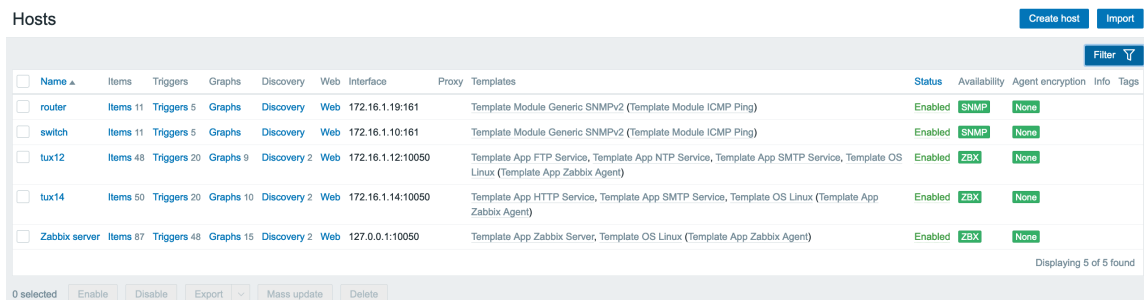
Figure 4.1

Após se assegurar a comunicação entre os diferentes sistemas, é iniciada a configuração da monitorização dos diferentes serviços.

Similarmente ao Nagios, e dependendo do hosts (Agent ou Agentless) e dos serviços nele alojados, os seguintes templates foram adicionados a cada um:

Sistema	Testes
Tux12	Template App Zabbix Agent Template App NTP Service Template App FTP Service Template App SMTP Service
Tux14	Template App Zabbix Agent Template App HTTP Service Template App SMTP Service
Router	Template Module Generic SNMPv2
Switch	Template Module Generic SNMPv2
Tux13	Template App Zabbix Server Template App Zabbix Agent

Table 4.1: Alocação dos serviços nos computadores

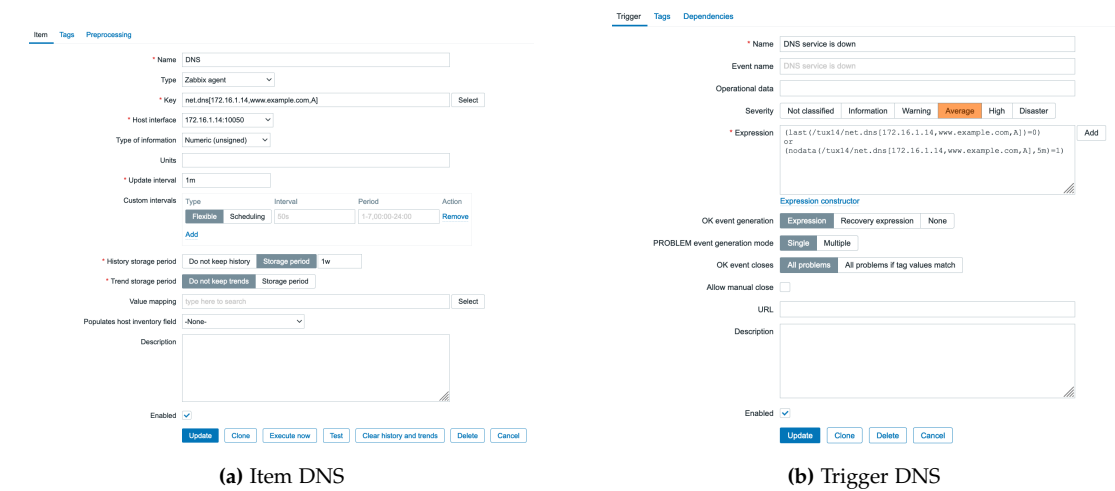


The screenshot shows the Zabbix 'Hosts' page. At the top right are buttons for 'Create host' and 'Import'. Below is a table with columns: Name, Items, Triggers, Graphs, Discovery, Web, Interface, Proxy, Templates, Status, Availability, Agent encryption, Info, and Tags. The table lists five hosts: 'router', 'switch', 'tux12', 'tux14', and 'Zabbix server'. Each host row shows its configuration details and the templates applied to it. For example, 'tux12' has templates for FTP, NTP, SMTP, and OS Linux. At the bottom, there are buttons for '0 selected', 'Enable', 'Disable', 'Export', 'Mass update', and 'Delete'. A status bar at the bottom right indicates 'Displaying 5 of 5 found'.

Name	Items	Triggers	Graphs	Discovery	Web	Interface	Proxy	Templates	Status	Availability	Agent encryption	Info	Tags
router	11	5	5	Discovery	Web	172.16.1.19:161		Template Module Generic SNMPv2 (Template Module ICMP Ping)	Enabled	SNMP	None		
switch	11	5	5	Discovery	Web	172.16.1.10:161		Template Module Generic SNMPv2 (Template Module ICMP Ping)	Enabled	SNMP	None		
tux12	48	20	9	Discovery	Web	172.16.1.12:10050		Template App FTP Service, Template App NTP Service, Template App SMTP Service, Template OS Linux (Template App Zabbix Agent)	Enabled	ZBX	None		
tux14	50	20	10	Discovery	Web	172.16.1.14:10050		Template App HTTP Service, Template App SMTP Service, Template OS Linux (Template App Zabbix Agent)	Enabled	ZBX	None		
Zabbix server	87	48	15	Discovery	Web	127.0.0.1:10050		Template App Zabbix Server, Template OS Linux (Template App Zabbix Agent)	Enabled	ZBX	None		

Figure 4.2: Templates associados a cada host

Como se pode observar, nenhum template associado ao serviço DNS foi adicionado. De facto, o Zabbix não fornece nenhum template default para esse fim. Embora seja possível importar templates externos, foi configurado um item e o correspondente trigger manualmente.



Trigger

Tags

Dependencies

Name

DNS service is down

Event name

DNS service is down

Operational data

Severity

Not classified

Information

Warning

Average

High

Disaster

Expression

(last(/usr14/net.dns[172.16.1.14,www.example.com,A])=0)
or
(nodata(/usr14/net.dns[172.16.1.14,www.example.com,A],5m)=1)

Add

Expression constructor

OK event generation

Expression

Recovery expression

None

PROBLEM event generation mode

Single

Multiple

OK event closes

All problems

All problems if tag values match

Allow manual close

☐

URL

Description

Enabled

☒

Update

Clone

Delete

Cancel

(b) Trigger DNS

Figure 4.3

Este teste verifica se a *query* feita ao servidor DNS é respondida. Se tal não se verificar, ou se não se obtiver resposta do host de todo, um alerta da gravidade *Average*, igual ao de outros serviços, é criado.

4.2 Resultados

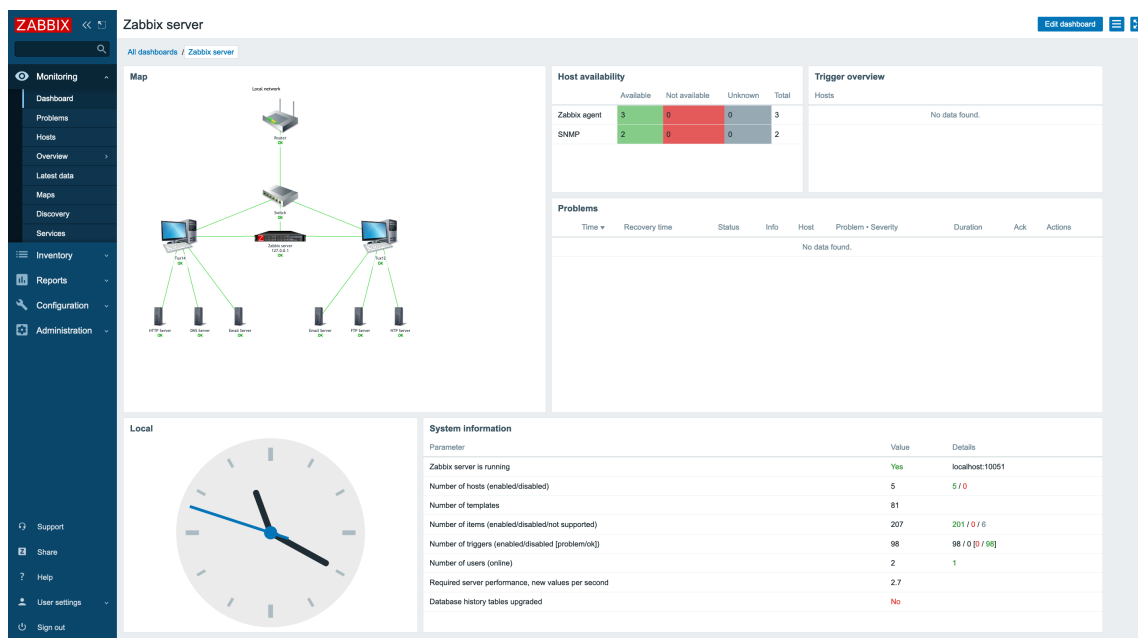


Figure 4.4: Dashboard principal

Uma das principais vantagens do Zabbix é o seu grau de customização, nomeadamente as **Dashboards**. Esta dashboard foi criada por nós, onde a informação mais relevante é apresentada:

- Hosts Availability: Quantos hosts então *up/down*.
- Trigger Overview: Quais foram os triggers que foram despoletados.
- Problems: Quais os problemas do rede.
- System Information: Compilação geral da configuração do sistema e status da rede.
- Local: Hora local definida no servidor, neste caso *Europe/Lisbon*.
- MAP: Interface que permite de uma forma visual verificar o status de serviços e hosts. Este mapa foi desenhado por nós para apresentar todos os hosts e os serviços neles alojados. É possivelmente a componente mais importante pois apresenta pragmaticamente e de forma simples toda a informação relevante neste trabalho.

Existem muitas outras interfaces com compilação de informação do sistema. É também possível ver a evolução do status de alguns parâmetros graficamente ou o histórico de outputs.

4.3 Teste de falhas

Os testes de falhas no Nagios e no Zabbix foram executados ao mesmo tempo, pelo que o método é o mesmo:

Tux12

```
systemctl stop vsftpd - Falha do servidor FTP
systemctl stop postfix - Falha do servidor Email
```

Tux14

```
systemctl stop bind9 - Falha do servidor DNS
systemctl stop apache2 - Falha do servidor HTTP
```

Após algum tempo de atualização de informação, o output da interface do Zabbix foi o seguinte:

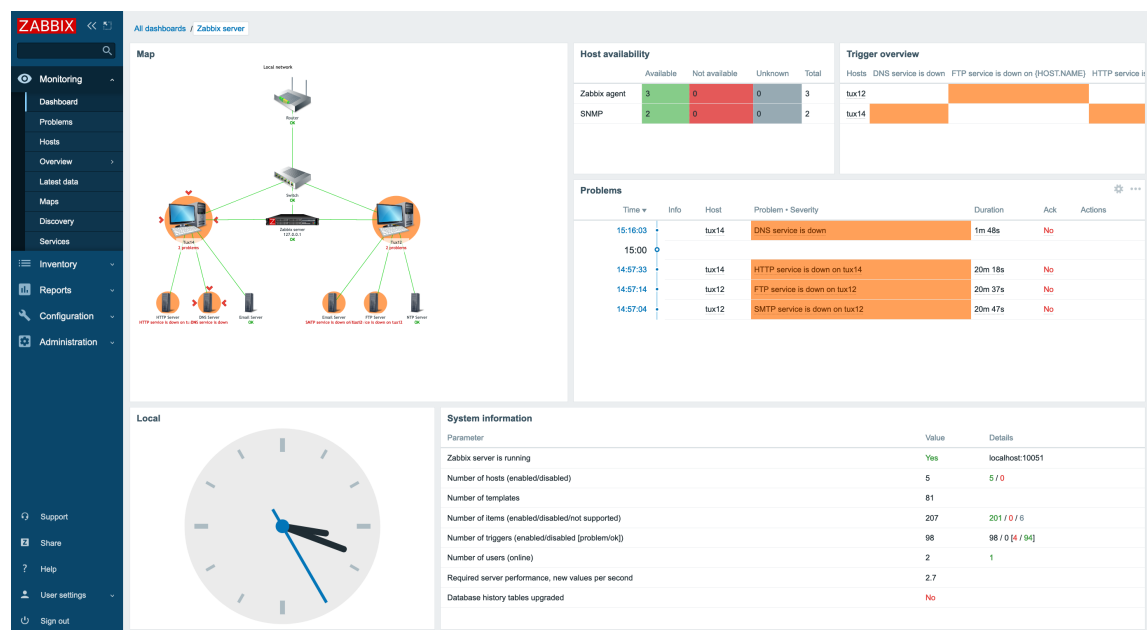


Figure 4.5: Falha de serviços

O Zabbix apresenta na tab *Problems* quais são os problemas encontrados, neste caso os serviços down, assim como o *timestamp* associado ao evento. É possível também observar quais foram os triggers despoletados, e em que hosts ocorreram. Por fim, de uma forma mais visual, os hosts onde há problemas são destacados com a cor do problema mais grave, neste caso de severidade *Average* que corresponde ao laranja.

Consequentemente, simulou-se a falha do Tux14, obtendo-se os seguintes resultados:

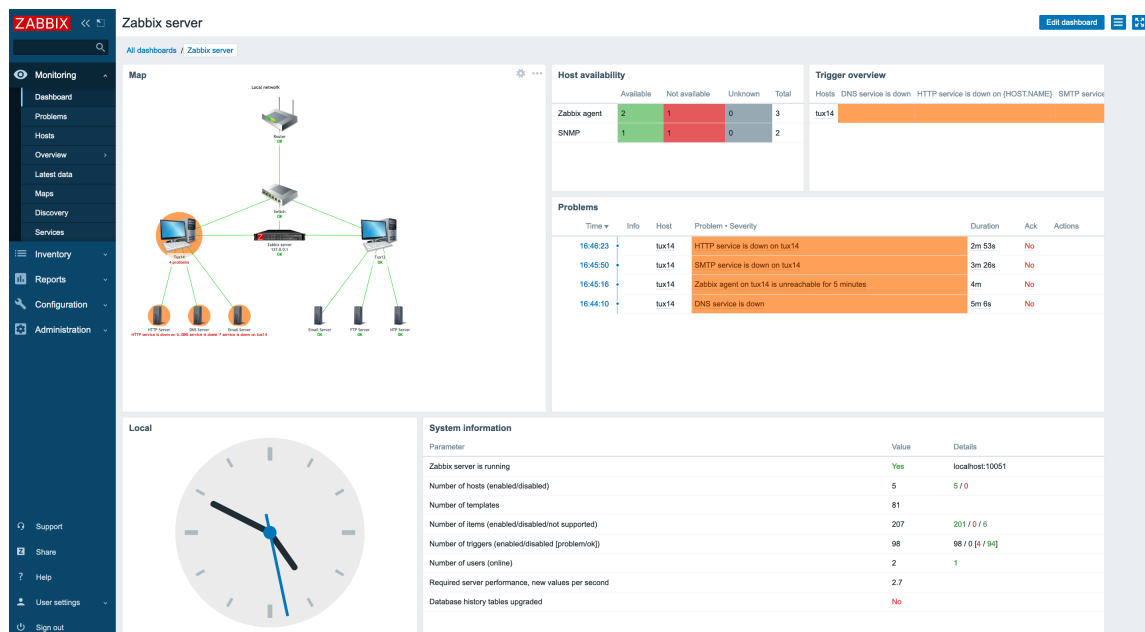


Figure 4.6: Falha do Tux14

É possível observar que o Zabbix indica que se perdeu contacto com o agente no Tux14. Desse modo, na tab *Host availability*, já se observa que um host está down. Os serviços alojados no host que falhou também falharam todos, como se pode ver quer no mapa, quer na tab *Problems*.

Sabendo que o host foi desligado às 16:40, o Zabbix demorou 4 minutos a detetar a falha no sistema, sendo que o estado serviços foi atualizado pouco depois. É de salientar que a frequência de teste pode ser alterado igualmente na configuração do Zabbix.

5 Outras ferramentas

5.1 Grafana

Grafana é uma ferramenta *open-source* de monitorização [11]. Esta ferramenta destaca-se pela sua interface UI muito interativa, apresentando vários gráficos e *gauges*. Deste modo, é possível criar dashboards muitos customizáveis e dinâmicas.

Ao contrário do Nagios e do Zabbix, este software é desenvolvido com um backend em *Go Lang*, uma linguagem mais recente e *higher-level* que *C Lang*.

Devido a ser uma ferramenta gráfica poderosa, é utilizada em complemento com ferramentas mais específicas mas não tão gráficas como o Zabbix, fornecendo assim informação high-level diretamente ao utilizador.



Figure 5.1: UI do Grafana

5.2 openDCIM

openDCIM é outra ferramenta de monitorização *open-source* orientada para *data centers* [12]. De facto, DCIM significa *Data Center Infrastructure Management*. É uma ferramenta mais específica, guardando informação relativa ao próprio hardware, sendo possível registar todo o inventário de equipamentos. É possível também realizar testes da própria infraestrutura de rede, simulando por exemplo um corte de energia e analisando os serviços afetados.

No entanto, a UI dessa desta ferramenta é muito mais simples e mais antiga, não sendo possível ter uma interface gráfica customizável como por exemplo com o Grafana. Disponibiliza também menos ferramentas de análise de serviços do que as outras opções.

The screenshot displays the openDCIM web interface for adding or editing a device. The form is organized into several panels. The 'Asset Tracking' panel on the left contains fields for Device ID (108), Status (Production), Label, Serial Number, Asset Tag, Primary IP / Host Name, Manufacture Date (1970-01-01), Install Date (2018-07-18), Warranty Company, Warranty Expiration (1970-01-01), Last Audit Completed (Audit not yet completed), Departmental Owner (RAD), and a 'Show Contacts' button. Below this is the 'Escalation Information' section with 'Time Period' and 'Details' dropdown menus. The 'Primary Contact' is set to 'Unassigned' and 'Tags' are 'Awaiting input...'. A 'Custom Attributes' section is at the bottom left. The right side of the form includes the 'Device Class' (Húsareyn / 4), physical dimensions (Height: 1, Position: 42), and configuration options (Half Depth checked, Back Side unchecked, Number of Data Ports: 1, Weight: 0, Power Connections: 2, Device Type: CDU). The 'SNMP Configuration' section shows SNMP Version (2c), Read Only Community, and Consecutive SNMP Failures (0). The 'Power Specifications' section at the bottom right includes Source Panel, Voltages, Breaker Size (# of Poles) (1), and Panel Pole Number. A red error message box is overlaid on the Label field, stating: '* This field is required * Minimum 3 characters allowed'.

Figure 5.2: UI do openDCIM

6 Análise comparativa

Após os testes dos diferentes serviços, é possível fazer uma comparação direta e determinar qual a melhor ferramenta em cada contexto.

Interface gráfica

O **Grafana** é a ferramenta que proporciona a melhor experiência gráfica ao utilizador. Como uma UI muito focada em gráficos temporais e *gauges*, é bastante fácil e intuitivo rapidamente analisar o estado do sistema.

O **Zabbix** no entanto também tem uma excelente interface, principalmente com a funcionalidade de construção de mapas que permitem esquematizar a rede e o seu estado. Além disso, fornece mais informação, principalmente low-level do que o **Nagios**.

Configuração

Neste ponto apenas é possível comparar as duas ferramentas usadas. A configuração no **Zabbix** é feita na sua interface Web, pelo que desse modo é muito mais intuitivo e simples modificar e personalizar a ferramenta. Apesar do **Nagios** também ser muito configurável, é preciso modificar os ficheiros .conf diretamente no sistema, o que torna a experiência mais complexa e menos intuitiva.

Outro fator prende-se com abordagem dos templates, que permite anexar rapidamente um conjunto de teste relevantes naquele contexto, por exemplo de um serviço HTTP ou FTP. O Nagios por outro lado, obriga a manualmente se configurar cada teste individualmente para cada host, sendo portanto um processo mais moroso.

Plugins vs Items/Triggers

Apesar da configuração do **Nagios** ser mais complexa, os plugins são mais configuráveis do que os items/triggers do **Zabbix**.

Quando se define um plugin, mais parâmetros podem ser definidos do que num item. O seu output pode também ser analisado mais precisamente, com intervalos de gravidade, por exemplo, de delay grave ou muito grave.

Além disso, é possível importar facilmente novos plugins como foi feito para o SNMP, enquanto que os items do Zabbix são predefinidos no próprio sistema.

Outras funcionalidades

Ambos as ferramentas apresentam a funcionalidade de **Autodiscovery**. Esta funcionalidade permite a descoberta de novos hosts que entraram na rede local. No entanto, apenas o Nagios XI (pago) tem esta funcionalidade por predefinição, pelo que desse modo é mais fácil configurar no Zabbix.

A nível de **protocolos de comunicação suportados**, distingui-se o facto do **Grafana** suportar mais tipos de bases de dados, nomeadamente em cloud como é o caso da AWS e Azure. Isto é sem dúvida um fator importante, principalmente para sistemas muito grandes onde é preciso grande capacidade de armazenamento.

Por fim, é possível, quer no Nagios, quer no Zabbix, **agendar manutenção**, isto é, *downtime* aceitável, aparecendo essa informação disponível na plataforma.

7 Conclusão

Neste trabalho, começamos por configurar diferentes serviços na bancada, nomeadamente servidores DNS, NTP, FTP, HTTP e Email. Nos dispositivos de rede, o switch e router, foi ativado o protocolo SNMP.

Procedeu-se à configuração quer do **Nagios**, quer do **Zabbix**. Estas ferramentas foram configuradas com o intuito de testar a disponibilidade quer dos próprios sistemas na rede, quer dos serviços neles alojados.

Concluiu-se que o processo de configuração no Zabbix é mais simples e intuitivo do que no Nagios. Isto deve-se ao facto da configuração no Zabbix ser feita na sua interface Web, ao ponto que no Nagios é feito diretamente editando ficheiros de configuração no sistema. No entanto, os **plugins** do Nagios provaram ser mais versáteis e diversificados do que os **items** do Zabbix, aumentando a especificidade dos testes executados na rede.

Após o teste de bom funcionamento dos serviços, foram provocadas falhas nos serviços e sistemas, e analisado os comportamentos das ferramentas. Ambas as ferramentas apresentaram a informação de forma clara.

Todavia, a interface do Zabbix é visualmente mais apelativa, devido à possibilidade de criar **dashboards** altamente customizáveis, onde o utilizador pode criar mapas interativos que esquematicamente representam a rede e o status dos seus sistemas e serviços.

Finalmente, fez-se uma pequena análise de outras duas ferramentas de monitorização.

O **Grafana** provou ser uma ferramenta de monitorização poderosa devido à sua UI baseada em gráficos temporais e *gauges*. Mostra assim ser uma boa ferramenta complementar para apresentar informação high-level ao utilizador.

O **openDCIM** é uma ferramenta vocacionada para monitorização de *data centers*, com algumas funcionalidades low-level não oferecidas por outras ferramentas. No entanto, fica atrás a nível de UI, assim como uso para generalizado para testes de serviços devido à sua fraca oferta de plugins.

Bibliografia

- [1] *Apache Homepage*. [Visitado 06-2021]. URL: \url{https://httpd.apache.org/}.
- [2] *Bind9 Homepage*. [Visitado 06-2021]. URL: \url{https://www.bind9.net/}.
- [3] *ntpd - Network Time Protocol (NTP) Daemon*. [Visitado 06-2021]. URL: \url{https://docs.ntpsec.org/latest/ntpd.html}.
- [4] *vsftpd Homepage*. [Visitado 06-2021]. URL: \url{https://security.appspot.com/vsftpd.html}.
- [5] *SNMP Configuration Guide Cisco*. [Visitado 06-2021]. URL: \url{https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/snmp/configuration/xe-16/snmp-xe-16-book/nm-snmp-cfg-snmp-support.html}.
- [6] *Postfix Ubuntu Documentation*. [Visitado 06-2021]. URL: \url{https://ubuntu.com/server/docs/mail-postfix}.
- [7] *Nagios Core*. [Visitado 06-2021]. URL: \url{https://www.nagios.org/projects/nagios-core/}.
- [8] *Nagios Documentation*. [Visitado 06-2021]. URL: \url{https://support.nagios.com/kb/article/nagios-core-installing-nagios-core-from-source-96.html#Debian}.
- [9] *Zabbix*. [Visitado 06-2021]. URL: \url{https://www.zabbix.com/}.
- [10] *Zabbix Documentation*. [Visitado 06-2021]. URL: \url{https://www.zabbix.com/documentation/current/manual}.
- [11] *Grafana*. [Visitado 06-2021]. URL: \url{https://grafana.com/}.
- [12] *openDCIM*. [Visitado 06-2021]. URL: \url{https://opendcim.org/}.