

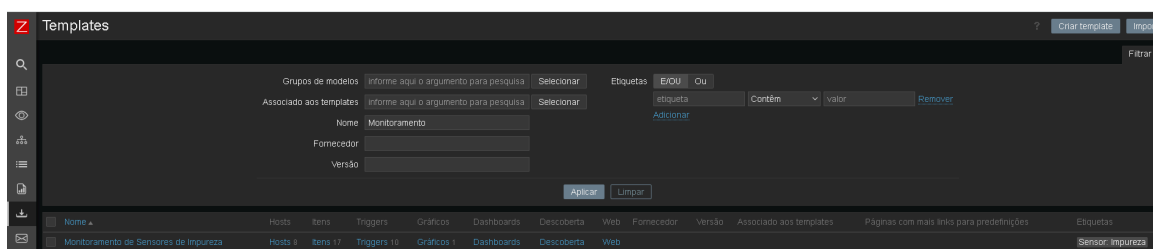


# Itens Zabbix

Itens no Zabbix são usados para coletar dados de um host. Após configurar um host, é necessário adicionar itens para iniciar a coleta de informações. Cada item representa uma métrica específica. Para facilitar, é possível criar templates com itens pré-definidos e vincular o host a um template para adicionar vários itens de uma vez.

## ▼ Itens padrões do Agent Zabbix

Temos um template padrão para monitoramento dos hosts de impureza. Nele adicionamos todos os itens de interesse e os hosts a serem coletados os dados.



O Zabbix possui diversos itens padrões para coleta de dados disponibilizados assim que é instalado o Agent Zabbix em um dispositivo. Nos usamos alguns desses itens padrões como:

→ *Estado do sistema; Memória Disponível; Memória Disponível %, Média de Carga da CPU (em 3 períodos diferentes); Quantidade de Processos Rodando; Tamanho Total do disco; Total de Memória; Total Disponível em Disco; Uso do Disco; Utilização de CPU; Número de CPUs; Horário no sensor e Tráfego de Rede (Entrada, Saída e Erros)*

Para adicionar um novo item pré-configurado, basta escolher uma das chaves disponíveis para o Agent Zabbix durante a criação de item no servidor Zabbix

## ▼ Adicionando itens padrões de hora local e tráfego de rede

### ▼ Hora local

Este item possibilita a coleta da data e hora do host. Para configurá-lo, basta selecionar a chave do item como `system.localtime[local]`, e a informação retornará no formato de texto.

Todos os hosts / sensor-test-rafael Ativo ZBX Itens 25 Triggers 10 Gráficos 1 Regras de descoberta Cenários web

Item Etiquetas 2 Pré-processamento

Itens herdados Monitoramento de Sensores de Impureza

\* Nome

Horário no sensor

Tipo

Agente Zabbix

\* Chave

system.localtime[local]

Tipo de informação

Texto

\* Interface do host

177.32.50.21:10050

\* Intervalo de atualização

10m

Intervalo customizado

Tipo

Flexível

Agendamento

50s

Período

1-7,00:00-24:00

Ação

Remover

Adicionar

\* Período de retenção do histórico

Não manter o histórico

Período de armazenamento

Preencha o campo do inventário do host

-Nenhum-

Descrição

Ativo

☒

Dados recentes

Atualizar

Clonar

Executar agora

Testar

Limpar histórico e estatísticas (médias)

Excluir

Cancelar

▼ Tráfego de rede

O Zabbix oferece vários itens padrão que auxiliam no monitoramento do tráfego de rede. Nós utilizaremos dois deles, `net.if.in[<parâmetros>]` e `net.if.out[<parâmetros>]`, para representar o tráfego de entrada e saída da interface eth0.

	Tipo	Agente Zabbix
<code>net.dns.record[&lt;ip&gt;,&lt;name&gt;,&lt;type&gt;,&lt;timeout&gt;,&lt;count&gt;,&lt;protocol&gt;]</code>		Executa uma consulta DNS. Retorna: texto com o tipo de informação requisitado
<code>net.dns[&lt;ip&gt;,&lt;name&gt;,&lt;type&gt;,&lt;timeout&gt;,&lt;count&gt;,&lt;protocol&gt;]</code>		Verifica se o serviço do DNS está em execução. Retorna 0 - DNS fora (servidor não respondeu ou a resolução de nome falhou); 1 - DNS em execução
<code>net.if.collisions[if]</code>		Número de colisões fora da janela. Retorna: número inteiro
<code>net.if.discovery</code>		Lista de interfaces de rede. Retorna: JSON
<code>net.if.in[if,&lt;mode&gt;]</code>		Retorna estatísticas de tráfego de uma interface. Retorna: número inteiro
<code>net.if.list</code>		Lista das interfaces de rede (incluindo o tipo da interface, status, endereço IPv4 e descrição). Retorna: texto
<code>net.if.out[if,&lt;mode&gt;]</code>		Estatísticas de saída de tráfego na interface de rede. Retorna: número inteiro
<code>net.if.total[if,&lt;mode&gt;]</code>		Sumarização das estatísticas de entrada e saída da interface de rede. Retorna: número inteiro
<code>net.tcp.listen[port]</code>		Verifica se determinada porta TCP está em modo de ESCUTA. Retorna: 0 - não está em modo de escuta; 1 - está em modo de escuta
<code>net.tcp.port[&lt;ip&gt;,&lt;port&gt;]</code>		Verifica se é possível estabelecer uma conexão TCP com a porta especificada. Retorna 0 - não foi possível conectar; 1 - Foi possível a conexão
<code>net.tcp.service.perf[service,&lt;ip&gt;,&lt;port&gt;]</code>		Verifica a performance de um serviço TCP. Retornos possíveis: 0 - serviço indisponível; <segundos> - tempo utilizado, em segundos, para se conectar ao serviço
<code>net.tcp.service[service,&lt;ip&gt;,&lt;port&gt;]</code>		Verifica se o serviço está em execução e aceitando conexões TCP. Retorna: 0 - serviço não está em execução; 1 - serviço em execução
<code>net.tcp.socket.count[&lt;laddr&gt;,&lt;lport&gt;,&lt;raddr&gt;,&lt;rport&gt;,&lt;state&gt;]</code>		Retorna o número de sockets TCP que correspondem aos parâmetros. Retorna um número inteiro

Para criar o item, escolha a chave `net.if.in[<interface>]` e especifique a interface desejada para monitoramento. Em seguida, preencha os próximos campos conforme necessário.

The screenshot shows the Zabbix web interface for creating a new item. The breadcrumb navigation at the top reads: "Todos os hosts / sensor-test-rafael Ativo ZBX Itens 25 Triggers 10 Gráficos 1 Regras de descoberta Cenários web". The "Itens" tab is active, and the "Pré-processamento 2" sub-tab is selected. The form contains the following fields and values:

- \* Nome:** Tráfego de entrada
- Tipo:** Agente Zabbix
- \* Chave:** net.if.in[eth0] (with a "Selecionar" button)
- Tipo de informação:** Numérico (inteiro sem sinal)
- \* Interface do host:** 177.32.50.21:10050
- Unidades:** bps
- \* Intervalo de atualização:** 10s
- Intervalo customizado:** A table with columns "Tipo", "Intervalo", "Período", and "Ação". It contains one row: "Flexível", "Agendamento", "50s", "1-7,00:00-24:00". There are "Adicionar" and "Remover" links.
- \* Período de retenção do histórico:** "Não manter o histórico", "Período de armazenamento", "1 d"
- \* Período de retenção das estatísticas:** "Não mantenha tendências", "Período de armazenamento", "365d"
- Mapeamento de valor:** informe aqui o argumento para pesquisa (with a "Selecionar" button)
- Preencha o campo do inventário do host:** -Nenhum-
- Descrição:** A large empty text area.

At the bottom left, there is a checkbox labeled "Ativo" which is checked.

Realize o mesmo processo para calcular o item de tráfego de saída, utilizando a chave `net.if.out[<interface>]`. Especifique a interface desejada para monitoramento e preencha os campos subsequentes conforme necessário.

Item Etiquetas Pré-processamento 2

\* Nome

Tipo

\* Chave

Tipo de informação

\* Interface do host

Unidades

\* Intervalo de atualização

Intervalo customizado

Tipo	Intervalo	Período	Ação
Flexível	Agendamento	50s	1-7,00:00-24:00

[Adicionar](#) [Remover](#)

\* Período de retenção do histórico  Período de armazenamento

\* Período de retenção das estatísticas  Período de armazenamento

Mapeamento de valor

Preencha o campo do inventário do host

Descrição

Ativo ☒

## ▼ Configurando comandos externos para buscar temperatura e ultimo resultado para itens personalizado

1. - Abra o arquivo `zabbix_agent2.conf` usando seu editor de texto preferido. Você pode fazer isso com o comando:

```
sudo nano /etc/zabbix/zabbix_agent2.conf
```

2. No final do arquivo `zabbix_agent2.conf`, adicione as seguintes linhas para permitir a execução dos comandos desejados:

```
AllowKey=system.run[curl -s http://192.168.10.1:8000/get_results]
AllowKey=system.run[vcgencmd measure_temp | sed -s "s/temp=//" | s
ed -s "s/'C//"]
```

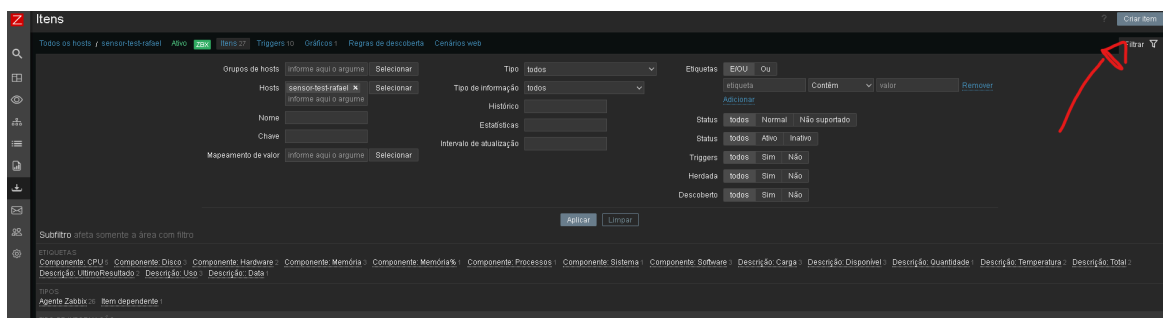
```
GNU nano 3.2 zabbix_agent2.conf
# 1 - enabled
#
# Mandatory: no
# Default:
# Plugins.SystemRun.LogRemoteCommands=0
## Option: ForceActiveChecksOnStart
# Perform active checks immediately after restart for first received configuration.
# Also available as per plugin configuration, example: Plugins.Uptime.System.ForceActiveChecksOnStart=1
#
# Mandatory: no
# Range: 0-1
# Default:
# ForceActiveChecksOnStart=0
# Include configuration files for plugins
Include=../zabbix_agent2.d/plugins.d/*.conf
AllowKey=system.run[python3 /home/pi/sensor/shell_scripts/logs_analytic.py]
AllowKey=system.run[curl -s http://192.168.10.1:8000/get_results]
AllowKey=system.run[vcgencmd measure_temp | sed -s "s/temp=//" | sed -s "s/'C//"]
|
```

3. Execute os seguintes comandos para recarregar os Daemons de serviços e reiniciar o agente Zabbix, para reconhecer as novas configurações.

```
sudo systemctl daemon-reload
```

```
sudo systemctl restart zabbix-agent2
```

4. Agora que o agente Zabbix foi configurado e reiniciado com sucesso, você pode adicionar um novo item no Zabbix Server.



5. Para adicionar o item de temperatura, preencha o formulário e escolha a chave `system.run[command,<mode>]` para executar o comando. Dentro dos colchetes, coloque o comando `vcgencmd measure_temp | sed -s "s/temp=//" | sed -s`

**"s"/"C"/"** exatamente como foi adicionado anteriormente no arquivo **zabbix\_agent2.conf**.

The screenshot shows the Zabbix web interface for configuring a new item. The breadcrumb trail at the top indicates the path: Todos os hosts / sensor-test-rafael / Ativo / ZBX / Itens 27 / Triggers 10 / Gráficos 1 / Regras de descoberta / Cenários web. The 'Item' tab is selected, and the 'Pré-processamento' sub-tab is active. The configuration form includes the following fields and options:

- Nome:** Temperatura
- Tipo:** Agente Zabbix
- Chave:** system.run[vcgencmd measure\_temp | sed -s "s/temp=/" | sed -s "s/C/" Selecionar
- Tipo de informação:** Numérico (fracionário)
- Interface do host:** 177.32.50.21:10050
- Unidades:** °C
- Intervalo de atualização:** 1m
- Intervalo customizado:** A table with columns Tipo, Intervalo, Período, and Ação. It contains one row: Flexível, Agendamento, 50s, 1-7,00:00-24:00, with a Remove button.
- Período de retenção do histórico:** Não manter o histórico, Período de armazenamento: 1d
- Período de retenção das estatísticas:** Não mantenha tendências, Período de armazenamento: 365d
- Mapeamento de valor:** Informe aqui o argumento para pesquisa, Selecionar
- Preencha o campo do inventário do host:** -Nenhum-
- Descrição:** A large text area for additional notes.
- Ativo:** Checked

At the bottom, there are buttons for 'Atualizar', 'Clonar', 'Executar agora', 'Testar', 'Limpar histórico e estatísticas (médias)', 'Excluir', and 'Cancelar'.

6. Preencha o formulário e escolha a chave **system.run[command,<mode>]** para adicionar o item do último resultado. Dentro dos colchetes, insira o comando **curl -s http://192.168.10.1:8000/get\_results**, exatamente como foi adicionado no arquivo **zabbix\_agent2.conf**.

Todos os hosts / sensor-test-rafael Ativo **ZBX** Itens 27 Triggers 10 Gráficos 1 Regras de descoberta Cenários web

Item Etiquetas 2 Pré-processamento 1

\* Nome Ultimo Resultado comando

Tipo Agente Zabbix

\* Chave system.run[curl -s http://192.168.10.1:8000/get\_results] Selecionar

Tipo de informação Texto

\* Interface do host 177.32.50.21:10050

\* Intervalo de atualização 1m

Intervalo customizado

Tipo	Intervalo	Período	Ação
Flexível	Agendamento	50s	1-7,00:00-24:00

Adicionar

\* Período de retenção do histórico Não manter o histórico Período de armazenamento 7d

Preencha o campo do inventário do host -Nenhum-

Descrição

Ativo ☒

Dados recentes

Atualizar Clonar Executar agora Testar Limpar histórico e estatísticas (médias) Excluir Cancelar

## ▼ Permitindo a execução de itens personalizados Zabbix

Os itens personalizados são ativados a partir do Agent Zabbix que pode executar scripts, realizar comandos e retornar uma resposta.

Para conseguirmos executar itens personalizados é necessário realizar algumas configurações no sensor.

1. Adicionar o serviço do Zabbix ao grupo de root:

```
sudo systemctl edit zabbix-agent2
```

2. Copie e cole as informações abaixo no edit:

```
[Service]
User=root
Group=root
```

3. Recarregue os Daemons e reinicie o serviço zabbix-agent:

```
sudo systemctl daemon-reload
```

```
sudo systemctl restart zabbix-agent2
```

2- Usaremos o nosso arquivo **UserParameter** para atrelar uma chave ao nosso script, deixando padronizado na seguinte estrutura de comando:

```
UserParameter=[nome_chave],[caminho_do_script_ou_comando]
```

Na pasta /etc/zabbix/zabbix\_agent2.d crie o arquivo UserParameter.conf

```
sudo nano /etc/zabbix/zabbix_agent2.d/UserParameter.conf
```

## ▼ Configurando scripts personalizados Python

No Zabbix, é possível executar scripts em Python pelo servidor através do agente. Neste tutorial, vou demonstrar como configurar o Zabbix Agent para receber comandos do Zabbix Server.

1- Abra o arquivo de configuração Zabbix\_agent2.conf

```
sudo nano /etc/zabbix/zabbix_agent2.conf
```

2- Ao final do arquivo, adicione a seguinte instrução para conceder permissão à execução do comando pelo Zabbix Server:

```
AllowKey=system.run[python3 /home/pi/sensor/python3_scripts/logs_analytic.py]
```



```

GNU nano 3.2 /etc/zabbix/zabbix_agent2.conf Modified
#
# Mandatory: no
# Default:
# Plugins.SystemRun.LogRemoteCommands=0
### Option: ForceActiveChecksOnStart
# Perform active checks immediately after restart for first received configuration.
# Also available as per plugin configuration, example: Plugins.Uptime.System.ForceActiveChecksOnStart=1
# Mandatory: no
# Range: 0-1
# Default:
# ForceActiveChecksOnStart=0
# Include configuration files for plugins
Include=../zabbix_agent2.d/plugins.d/*.conf
AllowKey=system.run[python3 /home/pi/sensor/python3_scripts/logs_analytic.py]

```

3- Adicione um novo item de coleta de dados, selecionando o tipo "Agente Zabbix" e configurando a chave como `system.run["python3 /caminho_do_script/script"]`. Isso permitirá a execução do script via Server e a captura da saída de texto como resultado.

The screenshot shows the Zabbix web interface for configuring a new item. The breadcrumb trail is 'Todos os hosts / sensor-test-rafael / Ativo / ZBX'. The main configuration form is as follows:

- Nome:** Análise de logs
- Tipo:** Agente Zabbix
- Chave:** system.run[python3 /home/pi/sensor/python3\_scripts/logs\_analytic.py] (with a 'Selecionar' button)
- Tipo de informação:** Texto
- Interface do host:** 177.32.50.21:10050
- Intervalo de atualização:** 1m
- Intervalo customizado:**

Tipo	Intervalo	Período	Ação
Flexível	Agendamento	50s	1-7,00:00-24:00

Buttons: Adicionar, Remover
- Período de retenção do histórico:** Não manter o histórico (selected) / Período de armazenamento: 90d
- Preencha o campo do inventário do host:** -Nenhum-
- Descrição:** (Empty text area)
- Ativo:** ☒

At the bottom, there are buttons for 'Atualizar', 'Clonar', 'Executar agora', 'Testar', 'Limpar histórico e estatísticas (médias)', 'Excluir', and 'Cancelar'.

## ▼ Scripts do sensor

### ▼ Diagnostico do sensor

Esse script verifica se o diagnostico esta rodando nos processos da Raspi. Primeiro filtrará por partes do texto, caso encontre, terá seu status de saída de sucesso.

1. Crie um script chamado `diagnosis_status.sh` e cole o código abaixo.

```
sudo nano sensor/shell_scripts/diagnosis_status.sh
```

```
#!/bin/bash
ps -aux | grep "python3 /home/pi/python3_scripts/start_diagnose.p
y" | grep "SI" >/dev/null

if [ $? -eq 0 ]; then
    echo 1
else
    echo 0
fi
```

2. De permissão para execução do script

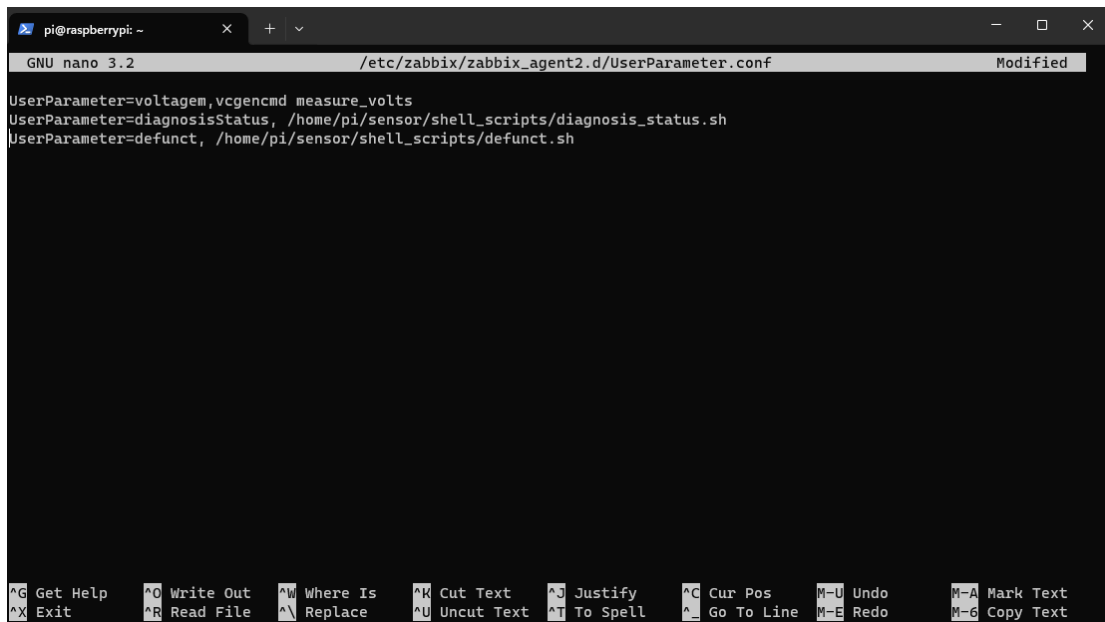
```
sudo chmod a+x sensor/shell_scripts/diagnosis_status.sh
```

3. Entre nas configurações do arquivo `UserParameter` do Zabbix para definir a chave e o caminho do script

```
/sudo nano /etc/zabbix/zabbix_agent2.d/UserParameter.conf
```

4. Adicione a seguinte chave no arquivo de `UserParameter` e salve-o

```
UserParameter=diagnosisStatus, /home/pi/sensor/shell_scripts/diag
nosis_status.sh
```



```
pi@raspberrypi: ~
GNU nano 3.2 /etc/zabbix/zabbix_agent2.d/UserParameter.conf Modified
UserParameter=voltage,vcgencmd measure_volts
UserParameter=diagnosisStatus, /home/pi/sensor/shell_scripts/diagnosis_status.sh
UserParameter=defunct, /home/pi/sensor/shell_scripts/defunct.sh

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos   M-U Undo     M-A Mark Text
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^G Go To Line M-E Redo     M-6 Copy Text
```

## 6. Recarregue os Daemons de serviços e reinicie o agente Zabbix

```
sudo systemctl daemon-reload
```

```
sudo systemctl restart zabbix-agent2
```

### ▼ Processos Zombies (Defunct)

Detecta processo Zumbi ativo filtrando partes do texto. Se identificados, retorna um status de saída bem-sucedido.

1. Crie um script chamado defunct.sh e cole o código abaixo.

```
sudo nano sensor/shell_scripts/defunct.sh
```

```
#!/bin/bash
ps -aux | grep "<defunct>" | grep "Z" >/dev/null

if [ $? -eq 0 ]; then
    echo 1

else
    echo 0
```

```
fi
```

2. De permissão para execução do script

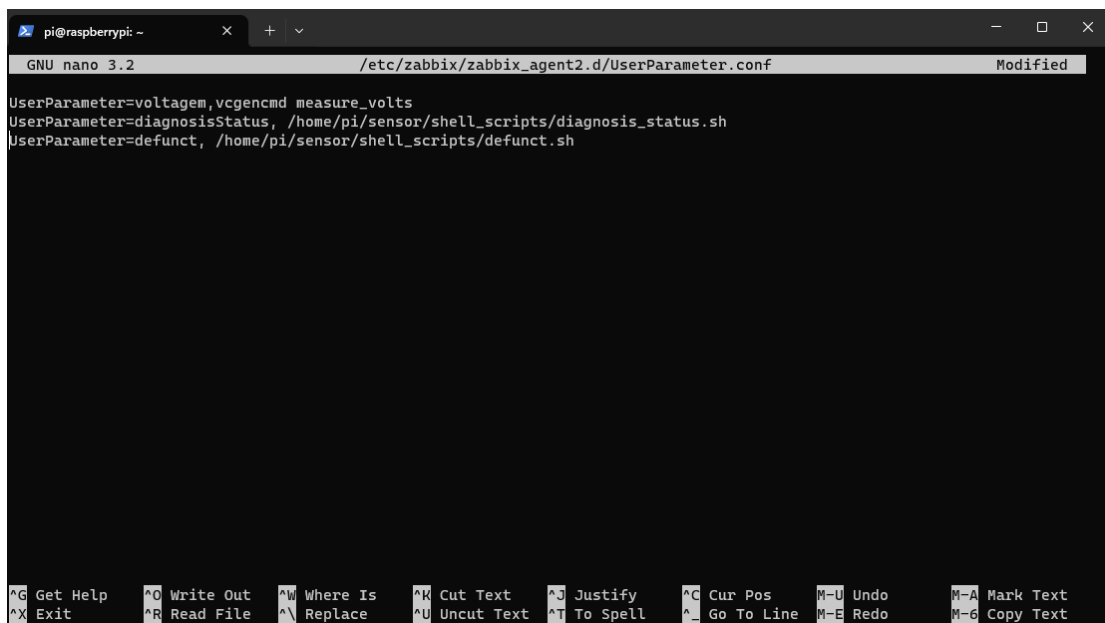
```
sudo chmod a+x sensor/shell_scripts/defunct.sh
```

3. Entre nas configurações do arquivo UserParmeter do Zabbix para definir uma chave e o caminho do script

```
/sudo nano /etc/zabbix/zabbix_agent2.d/UserParameter.conf
```

4. Adicione a seguinte chave no arquivo de UserParameter e salve-o

```
UserParameter=defunct, /home/pi/sensor/shell_scripts/defunct.sh
```



```
pi@raspberrypi: ~  
GNU nano 3.2 /etc/zabbix/zabbix_agent2.d/UserParameter.conf Modified  
UserParameter=voltagem,vcgencmd measure_volts  
UserParameter=diagnosisStatus, /home/pi/sensor/shell_scripts/diagnosis_status.sh  
UserParameter=defunct, /home/pi/sensor/shell_scripts/defunct.sh  
  
^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo     M-A Mark Text  
^X Exit      ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line  M-E Redo     M-6 Copy Text
```

6. Recarregue os Daemons de serviços e reinicie o agente Zabbix

```
sudo systemctl daemon-reload
```

```
sudo systemctl restart zabbix-agent2
```

▼ **Análise de requisições do sensor**

O script em Python foi desenvolvido para extrair dos logs atuais informações sobre os hosts e a quantidade de requisições realizadas, retornando uma lista de dicionários.

```
import json
import re

file_path = "/home/pi/sensor/logs/sentinel.log"

# Leitura do arquivo sentinel.log
def read_logs(log_file_path):
    with open(log_file_path, "r") as file:
        return file.readlines()

def extract_unique_ips(logs):
    ip_pattern = re.compile("\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}")
    return list(set(ip_pattern.findall(''.join(logs))))

def define_status_ranges():
    return {
        '2xx': range(200, 300),
        '4xx': range(400, 500),
        '5xx': range(500, 600)
    }

# Contar solicitações com base no código de status HTTP
def count_requests(logs, host, method, status_range):
    return sum(
        1 for log in logs if 'GET' in log and method in log and str(host) in log and int(log[-8: -4]) in status_range
    )

# Criar análise de solicitações para cada host e metodo
def create_request_analysis(logs, hosts, methods, status_ranges):
```

```

analysis_list = []
for host in hosts:
    for method in methods:
        counts = {f"{status_type}": count_requests(logs, host, method, status_ranges[status_type]) for status_type
                    in status_ranges}
        total_count = sum(counts.values())
        if total_count != 0:
            analysis = {
                "IP": host,
                "method": method,
                "count": total_count,
                **counts
            }
            analysis_list.append(analysis)
return analysis_list

```

```

def host_dictionary_creation(log_file_path):
    logs = read_logs(log_file_path)
    unique_ips = extract_unique_ips(logs)
    methods = ['/get_results', '/show_results', '/update', '/faq', '/download_take']
    status_ranges = define_status_ranges()
    return create_request_analysis(logs, unique_ips, methods, status_ranges)

```

# Ordenar lista por IP

```

def sort_analysis_list(analysis_list):
    return sorted(analysis_list, key=lambda x: x['IP'])

```

```

def print_analysis(analysis_list):
    print(json.dumps(analysis_list, indent=4))

```

```

def main(log_file_path):

```

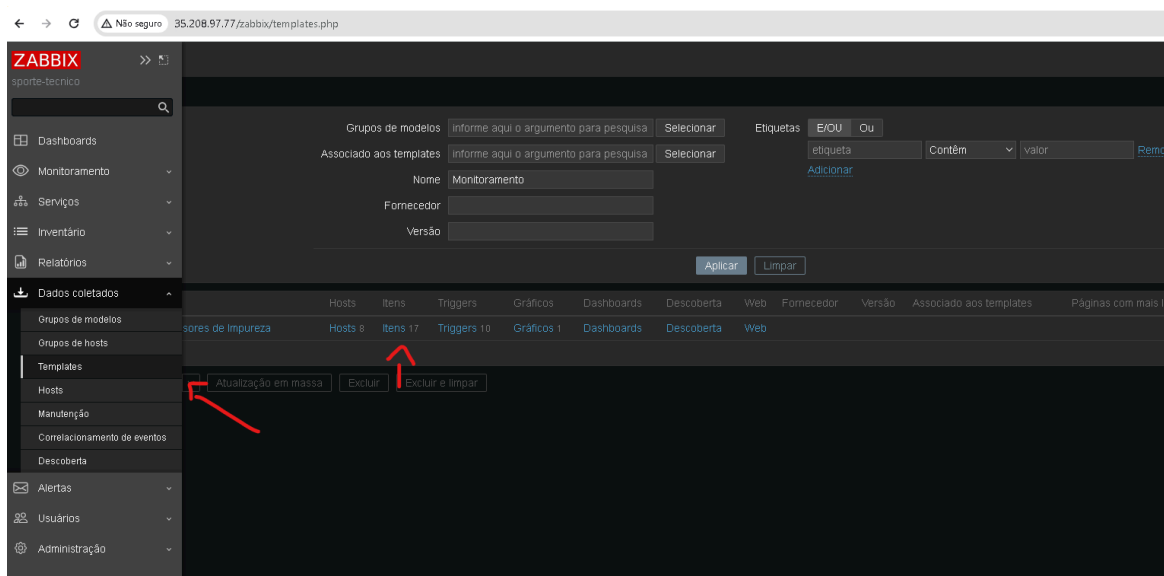
```
analysis_list = host_dictionary_creation(log_file_path)
sorted_analysis = sort_analysis_list(analysis_list)
print_analysis(sorted_analysis)
```

```
main(file_path)
```

## ▼ Adicionando itens personalizado para coleta dos scripts no Zabbix Server

Para adicionar um item no servidor Zabbix que coleta os dados dos scripts criados

1. basta ir na sessão templates e clicar na coluna itens



2. No canto superior direito, clique em "Criar Item"

**Itens**

Todos os templates / Monitoramento de Sensores / **Itens 17** / Triggers 10 / Gráficos 1 / Dashboards / Regras de descoberta / Cenários web

Grupos de modelos: Informe aqui o argumento | Selecionar

Temas: Monitoramento | Selecionar

Nome:

Chave:

Mapeamento de valor: Informe aqui o argumento | Selecionar

Tipo: todos

Etiquetas: EIOU | Ou

estatística: Contém | valor | Remover

Adicionar

Status: todos | Ativo | Inativo

Triggers: todos | Sim | Não

Herança: todos | Sim | Não

Aplicar | Limpar

Subfiltro afeta somente a área com filtro

ETIQUETAS: Componente: CPU | Componente: Disco | Componente: Hardware | Componente: Memória | Componente: Memória% | Componente: Processos | Componente: Sistema | Componente: Software | Descrição: Carga | Descrição: Disponível | Descrição: Quantidade | Descrição: Temperatura | Descrição: Total | Descrição: ÚltimoResultado | Descrição: Uso | Descrição: Data

TIPO: Agente Zabbix 16 | Item dependente 1

TIPO DE INFORMAÇÃO: Numérico (fracionário) | Numérico (inteiro sem sinal) | Texto 1

COM TRIGGERS: Com triggers 1 | Sem triggers 5

HISTÓRICO: 1d | 1h | 1m | 12h | 6d

ESTATÍSTICAS: 0 | 52s | 1d | 1h

INTERVALO: 30s | 1m | 1h | 10m | 15m

	Nome	Triggers	Chave	Intervalo	Histórico	Estatísticas	Tipo	Status	Etiquetas
...	Estado do sistema	Triggers 1	system uptime	30s	7d	0	Agente Zabbix	Ativo	Componente: Sist...
...	Horário no sensor		system.localtime[local]	10m	90d		Agente Zabbix	Ativo	Componente: Soft...   Descrição: Data
...	Memória Disponível	Triggers 1	vm.memory.size[available]	1m	7d	365d	Agente Zabbix	Ativo	Componente: Mem...   Descrição: Disponi...
...	Memória Disponível %		vm.memory.size[available]	1m	7d	365d	Agente Zabbix	Ativo	Componente: Mem...   Descrição: Disponi...
...	Média de Carga (1m em mib/s)	Triggers 1	system.cpu.load[all.avg1]	1m	7d	365d	Agente Zabbix	Ativo	Componente: CPU   Descrição: Carga

- Em seguida, preencha o formulário do item. No Campo Chave, coloque o mesmo nome atribuído ao script dentro do arquivo **UserParameter**. Escolha também qual o tipo do dado que será retornado no campo **Tipo de informação**.

Todos os hosts / sensor-test-rafael / Ativo **ZBX** / Itens 27 / Triggers 10 / Gráficos 1 / Regras de descoberta / Cenários web

**Item** | Etiquetas | Pré-processamento 1

\* Nome: Status do diagnostico

Tipo: Agente Zabbix

\* Chave: diagnosisStatus | Selecionar

Tipo de informação: Texto

\* Interface do host: 177.32.50.21:10050

\* Intervalo de atualização: 10m

Intervalo customizado

Tipo	Intervalo	Período	Ação
Flexível	Agendamento	50s	1-7,00:00-24:00

Adicionar

\* Período de retenção do histórico: Não manter o histórico | Período de armazenamento: 7d

Preencha o campo do inventário do host: -Nenhum-

Descrição:

Ativo: ☒

Dados recentes

Atualizar | Clonar | Executar agora | Testar | Limpar histórico e estatísticas (médias) | Excluir | Cancelar

Após o preenchimento, clique em atualizar.

- Faça o mesmo para o item defunct



Todos os hosts / sensor-test-rafael Ativo **ZBX** Itens 27 Triggers 10 Gráficos 1 Regras de descoberta Cenários web

Item Etiquetas Pré-processamento

\* Nome

Tipo

\* Chave  Selecionar

Tipo de informação

\* Interface do host

Unidades

\* Intervalo de atualização

Intervalo customizado

Tipo	Intervalo	Período	Ação
<input checked="" type="checkbox"/> Flexível	<input type="text" value="Agendamento"/>	<input type="text" value="50s"/>	<input type="text" value="1-7,00:00-24:00"/>
			<a href="#">Remover</a>
<a href="#">Adicionar</a>			

\* Período de retenção do histórico  Período de armazenamento

\* Período de retenção das estatísticas  Período de armazenamento

Mapeamento de valor  Selecionar

Preencha o campo do inventário do host

Descrição

Ativo ☒

[Dados recentes](#)

[zabbix\\_agent2.conf](#)