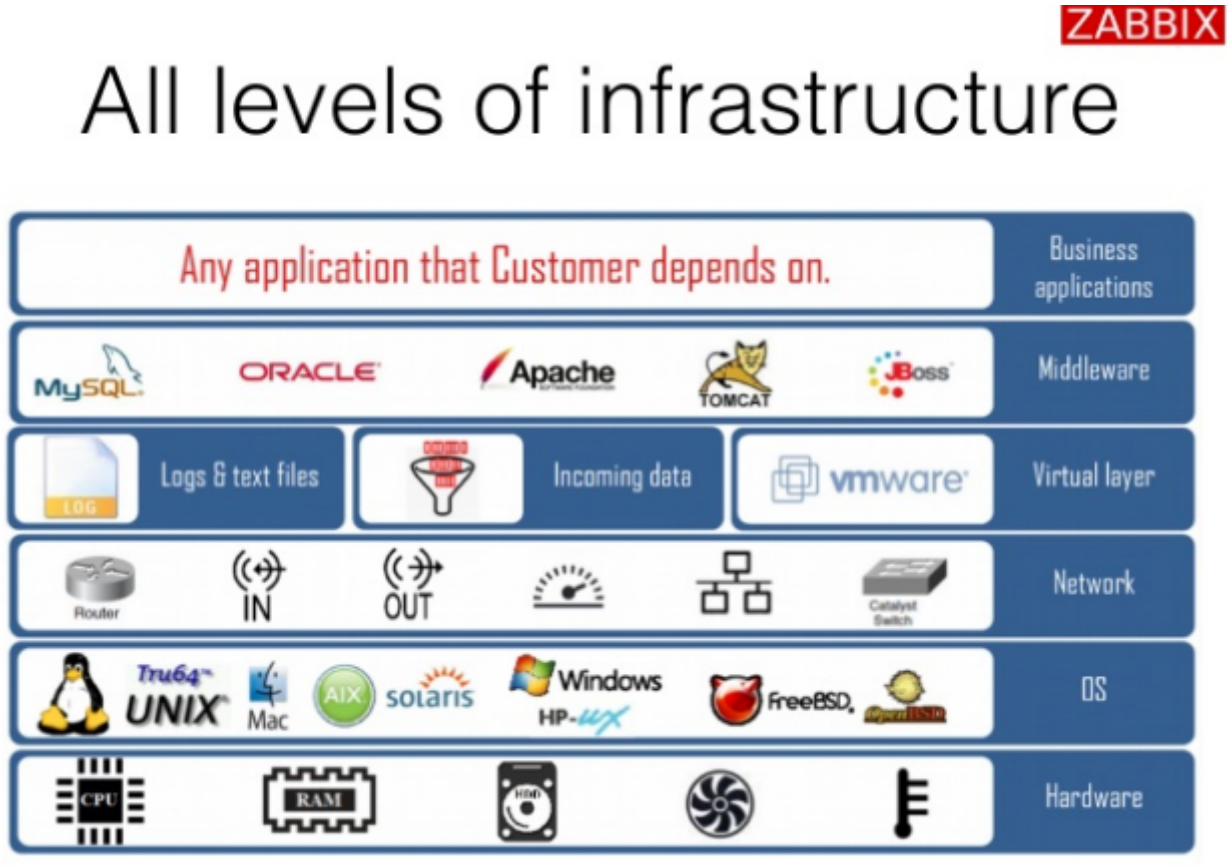
Zabbix + UserParameter + Expressões Regulares = Monitoramento otimizado

Amaury Souza
Sep 13, 2019 · 6 min read

Olá pessoal! Hoje o artigo será sobre **zabbix** (open-source monitoring software tool), no qual irei abordar um recurso importante, que é o **userparameter**. Atrelado à isso, vou exemplificar usando **expressões regulares** para automatizar o monitoramento da sua infra de TI.



Níveis de infraestrutura do Zabbix.

Considerarei que você tenha instalado um **zabbix-server** no seu ambiente de testes, pode ser em Docker, VirtualBox ou VMware, uma instância em uma nuvem pública ou privada ou uma máquina virtual provisionada no hipervisor (oVirt, Xen por exemplo).

O objetivo é mostrar de forma básica o uso desse recurso do **zabbix**, visto que, vejo muitos profissionais que ainda não utilizam tal funcionalidade. Vale lembrar que na versão 4.2 do **zabbix** já existe o recurso do **http agent**. Eu fiz uma instalação **all-in-one** e usarei o próprio host do **zabbix-server** para coleta dos dados.

- **O que será aprendido nesse artigo:**

> *Quando devemos utilizar Userparameter?*
> *Case: Monitorando MariaDB (processos, uptime, versão, queries, tasks e serviço);*
> *Criação de expressões regulares;*
> *Criação de aplicação e item;*
> *Verificação dos dados coletados no frontend.*

- **Quando devemos utilizar Userparameter?**

Esse recurso do **zabbix** é utilizado para estender as funcionalidades de acordo com o contexto da sua infra, ou seja, as vezes você precisa extrair um dado que não faz parte do complexo nativo do **zabbix**, sendo necessário nessa etapa a utilização do userparameter, passando um comando/expressão regular que colete um serviço, processo, arquivo do seu host, etc.

Vou deixar abaixo um link onde você pode consultar a documentação oficial do **zabbix**, para estudar mais sobre essa ferramenta. Zabbix Documentação.
Vamos prosseguir mostrando como monitorar de forma básica um database:

Case: Monitorando o MariaDB com userparameter.

Nessa parte, vou mostrar como se faz o uso do userparameter para coletar alguns dados do MariaDB usando **expressões regulares** e **encadeamento de comandos.** Os tópicos abaixo são partes desse case.

É possível utilizar templates para coletar os dados de um database, inclusive existe o **zabbix share,** onde você pode buscar por modelos de templates para vários tipos de **SGBD**, mas hoje não trataremos isso.
A ideia aqui é mostrar que o **userparameter** funciona bem.

Sintaxe do userparameter: `UserParameter=<key>,<command>`

No arquivo de configuração do **zabbix-agent** você deve ver o modelo abaixo, onde mostra a sessão do userparameter:

```
### Option: UserParameter
#       User-defined parameter to monitor. There can be several user-defined parameters.
#       Format: UserParameter=<key>,<shell command>
#       See 'zabbix_agentd' directory for examples.
#
# Mandatory: no
# Default:
#UserParameter
```

Arquivo de configuração Zabbix Agent.

O arquivo do **zabbix-agent** fica em `/etc/zabbix/zabbix_agentd.conf`

Vale lembrar que, toda vez que você alterar o arquivo do **zabbix-agent**, você deve reiniciar o serviço para que as alterações sejam feitas. O userparameter contém uma chave, nessa chave você adiciona um valor, que será a chave para a criação do item no **frontend** do **zabbix**.

Eu vou utilizar o banco de dados MariaDB que foi instalado com o **zabbix -server** aqui, e vou monitorar os (processos, uptime, versão, status das queries, tasks e o serviço do MariaDB) com userparameter.

- **Criação de expressões regulares para o userparameter.**

Nessa parte vamos listar o que queremos coletar e mostrar como utilizar as expressões regulares para nos ajudar nesse processo.

Para verificar algumas informações do banco de dados **MariaDB**, tenho que usar o comando abaixo, ele lista o status do database e mostra outros dados:

```
$ systemctl status mariadb
● mariadb.service - MariaDB 10.1.41 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled;
vendor preset: disabled)
  Drop-In: /etc/systemd/system/mariadb.service.d
           └─migrated-from-my.cnf-settings.conf
   Active: active (running) since Qui 2019-09-05 09:43:11 -03; 1h
19min ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
  Process: 20130 ExecStartPost=/bin/sh -c systemctl unset-
```

```
environment _WSREP_START_POSITION (code=exited, status=0/SUCCESS)
  Process: 20090 ExecStartPre=/bin/sh -c [ ! -e
/usr/bin/galera_recovery ] && VAR= ||
VAR=`/usr/bin/galera_recovery`; [ $? -eq 0 ]   && systemctl set-
environment _WSREP_START_POSITION=$VAR || exit 1 (code=exited,
status=0/SUCCESS)
  Process: 20088 ExecStartPre=/bin/sh -c systemctl unset-environment
_WSREP_START_POSITION (code=exited, status=0/SUCCESS)
 Main PID: 20102 (mysqld)
   Status: "Taking your SQL requests now..."
    Tasks: 44
   Memory: 146.1M
   CGroup: /system.slice/mariadb.service
           └─20102 /usr/sbin/mysqld
```

Analisando essa saída de comando, ele retorna um monte de informações, sendo que precisamos apenas de alguns dados, nesse caso a solução é começar com regex e encadeamentos de comandos, abaixo você deve ver como que ficará isso.

Temos que coletar (processos, uptime, versão, status das queries, tasks e o serviço do MariaDB).

- Vamos começar tratando a saída de processos do banco:

```
$ ps axu | grep -v grep | grep mysql | wc -l
1
```

- Já para o uptime do serviço ficaria assim:

```
$ systemctl status mariadb | grep Active | sed -n '/ago$/p' | sed
's/.\{62\}//'
1h 27min ago
```

- Para monitorar a versão vamos usar outro comando:

```
$ mysql -V
mysql  Ver 15.1 Distrib 10.1.41-MariaDB, for Linux (x86_64) using
readline 5.1
```

Vejam que a saída ficou extensa, eu quero exatamente a versão. Vou usar um comando muito útil que é o awk:

```
$ mysql -V | awk '/^mysql/ {print $1,$2,$3,$4,$5}'
  mysql Ver 15.1 Distrib 10.1.41-MariaDB,
```

- Agora vamos tratar o status das queries:

```
$ systemctl status mariadb | grep Status
  Status: "Taking your SQL requests now..."
```

- Já para as tasks podemos fazer de forma simples também:

```
$ systemctl status mariadb | grep -i tasks
  Tasks: 47
```

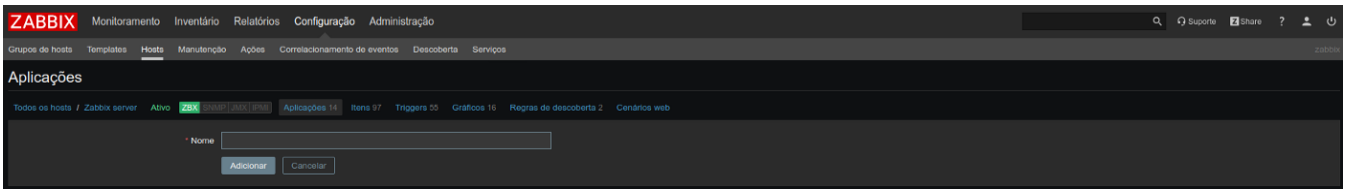
- Para finalizar, vamos utilizar alguns comandos para mostrar o serviço do MariaDB, o uso do pipe (|) é muito bom, dessa forma fica fácil a visualização depois no frontend do **zabbix**:

```
$ systemctl status mariadb | grep -i active | cut -d " " -f 4,5,6
  Active: active (running)
```

Com apenas algumas expressões regulares conseguimos simplificar os comandos. “Fácil como voar”!?

- **Criação de item.**

Nessa parte, como eu vou utilizar o próprio host do **zabbix-server,** vou criar uma aplicação chamada MariaDB.



Tela de aplicação do Zabbix.

> *Antes de criar o item no frontend, temos que testar o userparameter, só assim podemos prosseguir com o monitoramento. Esses passos são importantes para você continuar nesse artigo.*

Como já criamos as expressões regulares, temos que testar os comandos para depois adicionar isso no arquivo do **zabbix-agent**, seguindo aquela sintaxe que foi passada acima. Para testar, temos que usar um utilitário do **zabbix**, chamado **zabbix_get**.

```
UserParameter=<key>,<command>

$ zabbix_get -s 127.0.0.1 -p 10050 -k service
Active: active (running)
```

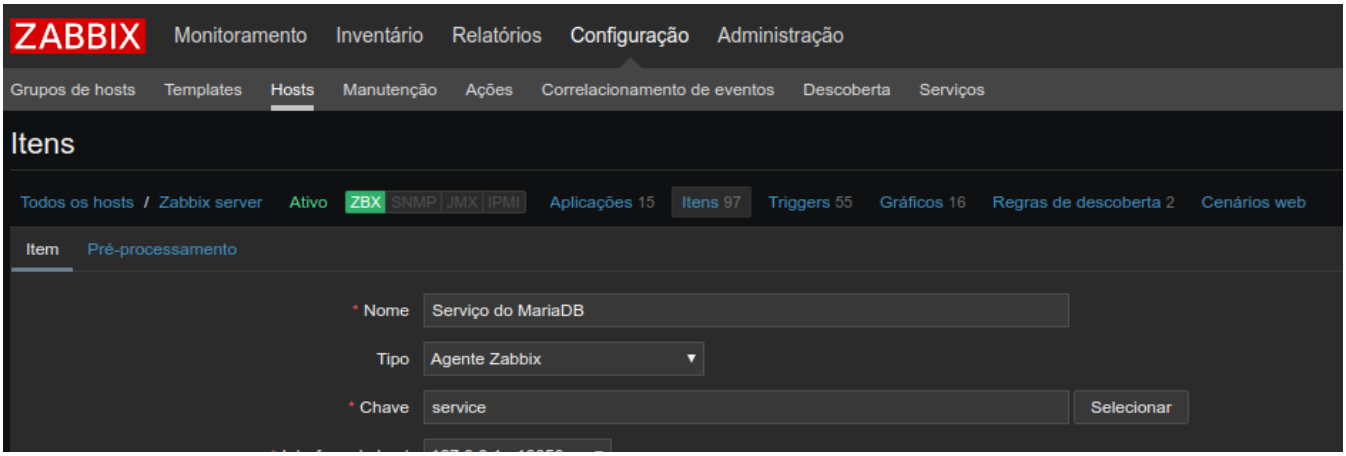
Utilizei os parâmetros do **zabbix_get**, que são, -s (IP), -k (chave) e -p (porta). O nosso userarameter está funcionando legal, podemos definir isso no frontend do **zabbix** criando nosso item agora:

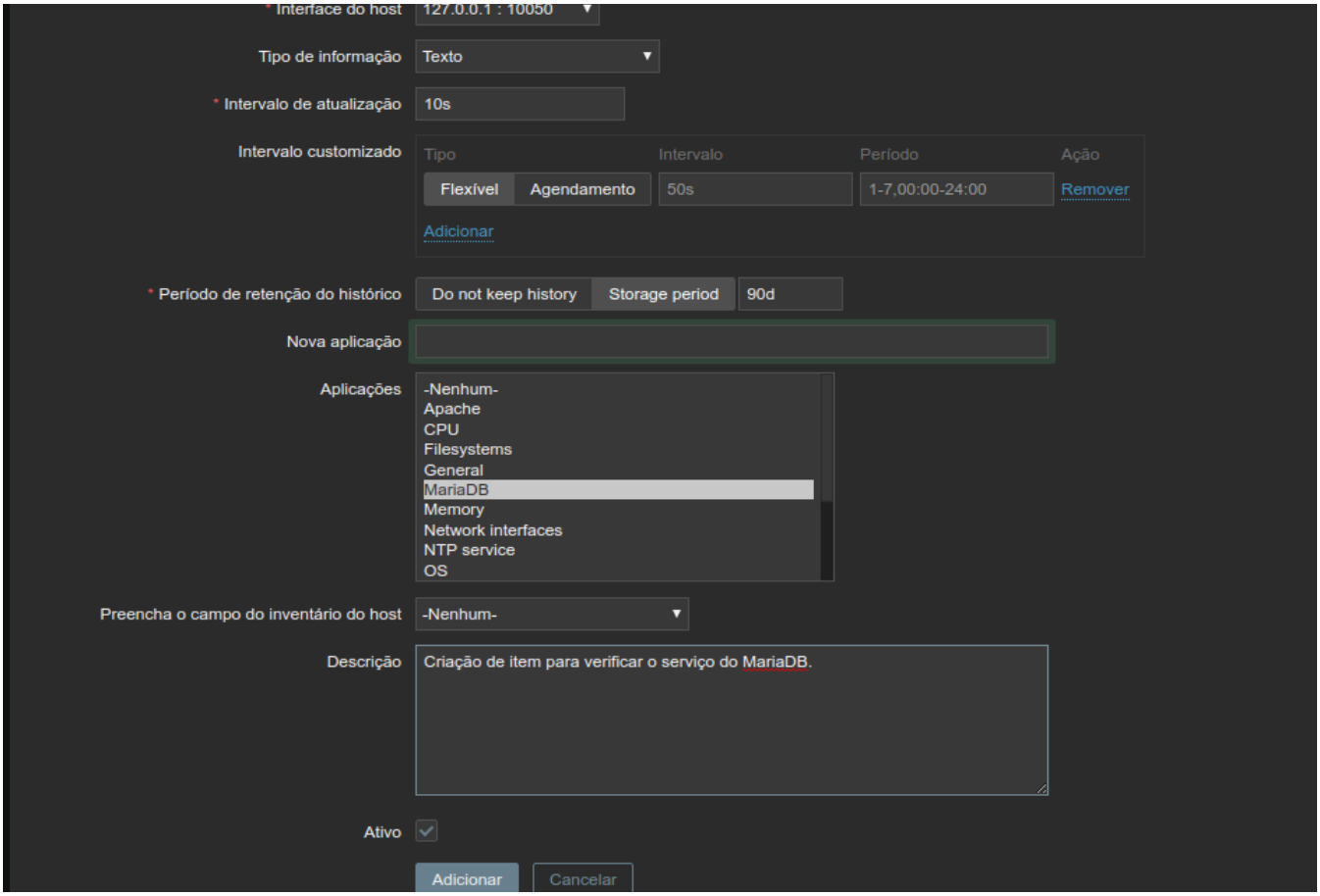
Veja como ficou meu arquivo:

```
UserParameter=process,ps axu | grep -v grep | grep mysql | wc -l
UserParameter=comando[*],du -sh $1
UserParameter=service,systemctl status mariadb | grep -i active | cut -d " " -f 4,5,6
UserParameter=tasks,systemctl status mariadb | grep -i tasks
UserParameter=status,systemctl status mariadb | grep Status
UserParameter=version,mysql -V | awk '/^mysql/ {print $1,$2,$3,$4,$5}'
UserParameter=uptime,systemctl status mariadb | grep Active | sed -n '/ago$/p' | sed 's/.\{62\}//'
```

Arquivo do zabbix-agent.

Vamos criar o item do serviço do MariaDB, onde eu coloco a chave que a gente criou, chamada de **service**:





Tela de criação de item

- **Verificação dos dados coletados no frontend.**

Olhando o frontend do **zabbix**, fica assim o monitoramento:

Dados recentes do zabbix.

Isso foi apenas para um item da nossa lista, que é o de serviços.

Para não estender muito esse artigo, irei adicionar o restante dos recursos, veja como ficou:

Monitoramento dos ativos do MariaDB.

Bom, esse foi um exemplo básico mesmo de como usar o recurso do **zabbix** (userparameter) com as expressões regulares, afim de otimizar o monitoramento de

determinado ativo da sua infra. Vale lembrar que na documentação oficial do **zabbix** você encontra outros recursos interessantes e até mesmo usar variáveis dentro de um userparamater. Eu não abordei o uso de triggers porque o artigo ficaria bem extenso, estou preparando um how-to apenas de **triggers** do **zabbix** para as próximas semanas.

Espero que tenham gostado do que foi comentado e fico à disposição sempre que vocês precisarem ;-) #VAIIII

Zabbix Infrastructure Userparameter