ZABBIX '24 CONFERENCE

LATIN AMERICA

JUNE, 07-08, 2024. SÃO PAULO, BRAZIL

ZABBIX '24
CONFERENCE
LATIN AMERICA

BIBLIOTECA ZABBIX_UTILS, HISTORY.PUSH E NOVIDADES DA API DO ZABBIX

BIBLIOTECA zabbix_utils

- Versão v1.0.0 criada em 17 de Novembro de 2023
- Anúncio oficial em 1º de Fevereiro de 2024 na versão v1.1.0

O que é? Para que serve? Como funciona?

- O que é?
  - ✓ Uma biblioteca em Python, OFICIALMENTE suportada pela nossa equipe de Integração (Andrey Biba e Aleksandr Iantsen)

  - ✓ Porque Python?
- Para que serve?
  - ✓ Simplificar a interação com a API do Zabbix, com o Zabbix server/proxy, e agent/agent2

BIBLIOTECA zabbix_utils

ZABBIX '24 CONFERENCE LATIN AMERICA

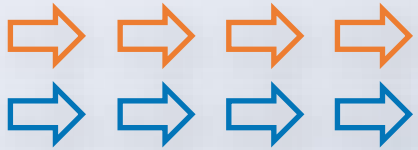
- Como funciona?

  A biblioteca pode operar em duas modalidades:

  ✓ Modo Síncrono (Sync):

  - Operações são executadas de forma sequencial;
  - Cada operação aguarda a conclusão da anterior;
  - Adequado para tarefas que exigem uma execução passo a passo.

  ✓ Modo Assíncrono (Async):



  - Operações são executadas de forma paralela;
  - Permite gerenciar operações concorrentes de forma eficiente;
  - Ideal para lidar com grande volume de operações ou operações de longa execução;
  - Adequado para ambientes de alto volume e concorrência;
  - Aumento de desempenho e escalabilidade de recursos.

BIBLIOTECA zabbix_utils

Instalação e configuração

- É possível realizar a instalação de duas maneiras
  - ✓ Utilizando PIP:

```
$ pip install zabbix_utils
$ pip install zabbix_utils[async] #Para o modo Assíncrono
```

  - ✓ Realizando clone do GitHub

```
$ git clone https://github.com/zabbix/python-zabbix-utils
$ cd python-zabbix-utils/
$ python setup.py install
```

- Versões suportadas e testadas:
  - ✓ Zabbix 5.0+
  - ✓ Python 3.8+

BIBLIOTECA zabbix_utils

ZABBIX '24 CONFERENCE LATIN AMERICA

zabbix_utils com a API do Zabbix

- Importar a classe **ZabbixAPI** da biblioteca e passar os devidos parâmetros:

```
from zabbix_utils import ZabbixAPI
api = ZabbixAPI(url="127.0.0.1",user="Admin",password="zabbix")
```

- Ao utilizar tokens é possível chamar o método login() para autenticação:

```
from zabbix_utils import ZabbixAPI
api = ZabbixAPI(url="127.0.0.1")
api.login(token="MEU_TOKEN_DAORA_EM_2D")
```

- Uma vez realizada a autenticação requests da API podem ser realizadas da seguinte forma:

```
api.host.get()
api.logout()
```

BIBLIOTECA zabbix_utils

Exemplos e comparação cURL vs zabbix_utils

```
curl --request POST \
  --url 'http://127.0.0.1/zabbix/api_jsonrpc.php' \
  --header 'Content-Type: application/json-rpc' \
  --header 'Authorization: Bearer 3770142786bbe04d574f5bd5d935aa29' \
  --data '{"jsonrpc":"2.0","method":"host.get","params":{"filter":{"host":"Zabbix
server"}},"id":1}'
```

```
from zabbix_utils import ZabbixAPI
api = ZabbixAPI(url="127.0.0.1/zabbix",user="Admin",password="zabbix")
host = api.host.get(
        filter={
            "host":"Zabbix server"
            }
        )
print(host)
```

BIBLIOTECA zabbix_utils

zabbix_utils com Getter

- Possibilita extrair informações do Zabbix agent/agent2
- Realizar chamadas para utilitários externos pode não ser muito conveniente
- Mesma ideia do zabbix_get
- Suporta as mesmas versões do agent suportadas pela API

```
from zabbix_utils import Getter

agent = Getter('10.50.0.71',10050)
resp = agent.get('system.uname')
```

BIBLIOTECA zabbix_utils

zabbix_utils com Getter

- O resultado do agent será processado pela biblioteca e retornado como um objeto da classe AgentResponse

```
print(resp)
# {
#     "error": null,
#     "raw": "Linux zabbix_server 5.15.0-
3.60.5.1.el9uek.x86_64",
#     "value": "Linux zabbix_server 5.15.0-
3.60.5.1.el9uek.x86_64"
# }

print(resp.error)
# None

print(resp.value)
# Linux zabbix_server 5.15.0-3.60.5.1.el9uek.x86_64
```

BIBLIOTECA zabbix_utils

zabbix_utils com Sender para server/proxy

- zabbix_sender é utilizado para enviar valores para o Zabbix trappers

```
zabbix_sender  -z  127.0.0.1  -s  “dummy host”  -k  miha.chave -o  50
```

- Como alternativa a classe Sender foi desenvolvida

```
from zabbix_utils import Sender
sender = Sender(server=“127.0.0.1”,port=10051)
resp = sender.send_value(‘dummy host’,’minha.chave’,50,
1714764660)
```

BIBLIOTECA zabbix_utils

zabbix_utils com Sender para server/proxy

- Alternativamente, é possível colocar agrupar os dados a serem enviados
- Para isso é necessário importar ItemValue

```
from zabbix_utils import ItemValue, Sender
items = [
    ItemValue('dummy1', 'item.key1', 10),
    ItemValue('dummy1', 'item.key2', 'Test value'),
    ItemValue('dummy2', 'item.key1', -1, 1702511920),
    ItemValue('dummy3', 'item.key1', '{"msg":"Test value"}'),
    ItemValue('dummy2', 'item.key1', 0, 1702511920, 100)
]

sender = Sender(server="127.0.0.1",port=10051)
resp = sender.send(item)
```

MÉTODO history.push

- O zabbix_sender precisa ser instalado
- A biblioteca zabbix_utils funciona em Python
- O método history.push facilita a integração com webhooks de terceiros de forma simplificada

Parâmetro	Tipo	Descrição
Itemid	ID	ID do item relacionado (obrigatório caso *host* e *key* não sejam definidos)
host	string	Nome técnico do host (obrigatório caso *itemid* não for definido)
key	string	Chave do item (obrigatório caso *itemid* não for definido)
value	Mixed	Valor do item (obrigatório)
clock	Timestamp	Hora quando o valor foi recebido.
ns	Integer	Nanossegundos de quando o valor foi recebido

MÉTODO history.push

- O zabbix_sender precisa ser instalado
- A biblioteca zabbix_utils funciona em Python
- O método history.push facilita a integração com webhooks de terceiros de forma simplificada

```
zabbix_sender  -z  zabbix.exemplo.com  -s  “Dummy”  -k  chave -o  10
```

```
curl --request POST \
  --url 'http://zabbix.exemplo.com/api_jsonrpc.php' \
  --header 'Authorization: Bearer
f191f8b1f47bc5ba5d1c0e6a91489503dd66e4abf486553c6913f34cec2a34d2' \
  --header 'Content-Type: application/json-rpc' \
  --data
'{"jsonrpc":"2.0","method":"history.push","params":{"host":"dummy","k
ey":"chave","value":10},"id":1}'
```

ZABBIX '24
CONFERENCE
LATIN AMERICA

Obrigado!