ZABBIX

zabbix_utils
Reminder and What's New

Aleksejs Abrosimovs

Intergration engineer

Who am I and what I'm doing here?

My name is Aleksejs Abrosimovs

- Intergration engineer
- About a year in Zabbix
- Still learning

I’m here to tell you about zabbix_utils

- Remind you of it
- Show you the progress
- Tell what’s new

What is zabbix_utils?

Zabbix official Python library

- **Working with Zabbix API**
  - Server
  - Proxies
  - Agents
- **Three parts (classes)**
  - ZabbixAPI
  - Getter
  - Sender
- **Answer to community and internal needs**
  - Compatibility with community libraries
  - Heavily used in internal tools

Sender

ZABBIX

Pushing Custom Metrics Like a Pro

- **Better alternative to CLI zabbix_sender**
  - Error handling
  - More options
- **Real-Time Data**
  - Push custom metrics as events happen.
  - Use your own super-custom schedule. Your choice, your preference.
- **Integration**
  - Embed monitoring directly into your apps.
  - Send metrics from your app to the Zabbix server. How cool is that?
- **Ease of Use**
  - Write clean, concise code to send data.
  - Create portable and reusable code.

Sending a custom metric

```python
from zabbix_utils import Sender

# Setup the sender with your Zabbix server details
sender = Sender(server='192.168.0.1', port=10051)

# Here we send a value '42' for the key 'custom.metric' from 'MyHost'
response = sender.send_value('MyHost', 'custom.metric', 42)

print("Metric sent!")

```

- Integration into your app / solution / workflow
  - Quickly
  - Effortlessly
  - Simple
  - Readable

Getter

Real-time value from an agent without going through the server

- **Better alternative to CLI zabbix_get**
  - Error handling
  - More options
- **Instant Feedback**
  - Grab the current state of an item directly from the agent.
- **Our own solution**
  - Your monitoring idea alongside with Zabbix
  - Or just quick custom check
- **Gather and analyze**
  - Collect what you need
  - Juggle the data as you wish

Pulling data directly from agents

```python
from zabbix_utils import Getter

# Connect directly to the Zabbix agent by its IP
getter = Getter(host='192.168.0.1', port=10050)
response = getter.get('system.uname')

print('System: ', response.value)

```

- Use agent for your own good
- Debug, pull metrics
- Collect

ZabbixAPI

Module handles all boring stuff for you, helps you.

- **Convenience**
  - No raw HTTP requests
  - No complicated JSON-RPC payloads
- **No limit in functionality**
  - Everything in GUI can be done via ZabbixAPI as well
  - And more
- **Less mistakes**
  - Error handling in responses
  - Syntax highlight and methods suggestion

Deleting the history of some items and printing users.

ZABBIX

- Import the class
- Create an object
  - With URL to your server
- “login” method
- “clear” method for history
- Get users
  - Use “get” for “user”
  - Filter what to get
- Print the result
- **Important!**
  “logout” method

```python
from zabbix_utils import ZabbixAPI

# Connect to the Zabbix server
zapi = ZabbixAPI(url='192.168.0.1')
zapi.login(user='Admin', password='zabbix')

# Delete history for items
ITEM_IDS = [70060, 70061, 70062]
zapi.history.clear(*ITEM_IDS)

# List users
users = zapi.user.get(output=['userid','name'])
for user in users:
    print(user['name' ])

zapi.logout()

```

cURL variant

```
1   #!/bin/bash
2
3   # login
4   curl --request POST --url 'http://127.0.0.1/api_jsonrpc.php' --header 'Content-Type:
    application/json-rpc' --data '{"jsonrpc":"2.0","method":"user.login","params":
    {"username":"Admin","password":"zabbix"},"id":1}'
5
6   # delete history
7   curl --request POST --url 'http://127.0.0.1/api_jsonrpc.php' --header 'Content-Type:
    application/json-rpc' --header 'Authorization: Bearer 52c73babb1cd1d0f24402932137415f0' --data
    {"jsonrpc":"2.0","method":"history.clear","params":["70060","70060","70060"],"id":1}
8
9   # get users
10  curl --request POST --url 'http://127.0.0.1/api_jsonrpc.php' --header 'Content-Type:
    application/json-rpc' --header 'Authorization: Bearer 52c73babb1cd1d0f24402932137415f0' --data
    '{"jsonrpc":"2.0","method":"user.get","params":{"output":["userid","username"]},"id":1}'
11
12  # logout
13  curl --request POST --url 'http://127.0.0.1/api_jsonrpc.php' --header 'Content-Type:
    application/json-rpc' --header 'Authorization: Bearer 52c73babb1cd1d0f24402932137415f0' --data
    '{"jsonrpc":"2.0","method":"user.logout","params":[],"id":1}'
14
```

Why should you care?

- **Automate repetitive tasks**
  - Some maintenance
  - User management
  - Import/export
- **Integrate Zabbix monitoring**
  - Befriend bidirectionally your app with Zabbix
- **Quickly debug and troubleshoot**
  - Sender
  - Getter
- **Focus on solving problems**
  - Think about big picture
  - Library will handle low level stuff

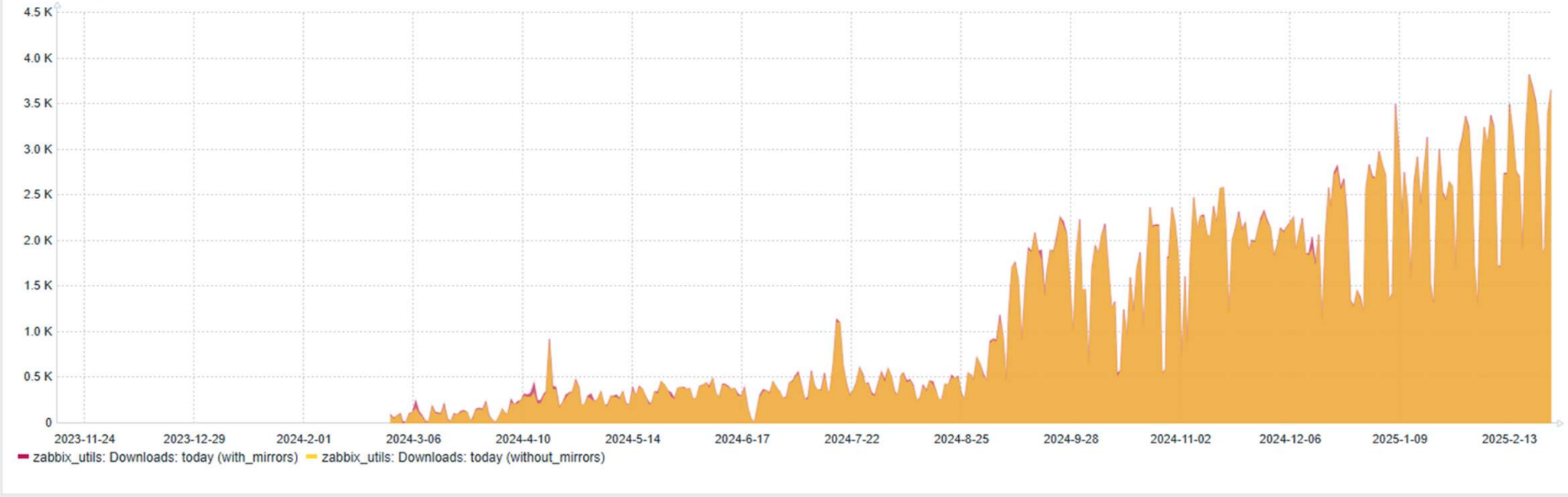
Progress since introduction

Achievements, milestones, dynamic

- **PyPI download stats**
  - **~ 80k** in the last month (**~ 2k** a year ago)
  - **~ 370k** overall downloads (**~ 7k** a year ago)
- **GitHub Engagement**
  - ~ 130 stars
  - ~ 23 forks
- **Growth**
  - Fast-paced by itself
  - Boosts from public mentions
  - Industry seasonal ups and downs

All time progress graph

Downloads: total (with_mirrors)	Downloads: total (without_mir...)	Downloads: last_month	Downloads: last_week	Downloads: last_day
368577	366144	80600	21377	3373



Changelog for the past year

- **Install from Zabbix Repository Packages**
  - For Red Hat / Fedora family:

```
# dnf install python3-zabbix-utils
```

  - For Debian / Ubuntu family:

```
# apt install python3-zabbix-utils
```

- **Support for Proxy Groups in Sender**

Changelog for the past year

ZABBIX

- **Zabbix Clusters Support in Sender**

```python
from zabbix_utils import ItemValue, Sender

# Create an instance of Sender specifying a list of Zabbix clusters:
zabbix_clusters = [
    ['zabbix.cluster1.node1', 'zabbix.cluster1.node2:10051'],
    ['zabbix.cluster2.node1:10051', 'zabbix.cluster2.node2:20051', 'zabbix.cluster2.node3']
]
sender = Sender(clusters=zabbix_clusters)

# List of ItemValue instances representing items to be sent
items = [
    ItemValue('host1', 'item.key1', 10),
    ItemValue('host1', 'item.key2', 'test message'),
    ItemValue('host2', 'item.key1', -1, 1695713666),
    ItemValue('host3', 'item.key1', '{"msg":"test message"}'),
    ItemValue('host2', 'item.key1', 0, 1695713666, 100)
]

# Send multiple items to the Zabbix server/proxy and receive response
response = sender.send(items)

```

Changelog for the past year

- **SSL connection in ZabbixAPI:**
  - ssl_context argument

```python
import ssl
from zabbix_utils import ZabbixAPI

# Create a SSL context and load a custom certificate
ctx = ssl.create_default_context()
ctx.load_verify_locations("/path/to/certificate.crt")

# Create an instance of the ZabbixAPI class
ZABBIX_AUTH = {
    "url": "https://example.com",
    "user": "Admin",
    "password": "zabbix",
    "ssl_context": ctx
}

# Login to the Zabbix API using provided user credentials
api = ZabbixAPI(**ZABBIX_AUTH)

# Retrieve a list of hosts and print their names
hosts = api.host.get(output=['hostid', 'name'])
for host in hosts:
    print(host['name'])

# Logout to release the Zabbix API session
api.logout()

```

Changelog for the past year

- **Asynchronous Modules (classes) were Introduced:**
  - AsyncZabbixAPI
  - AsyncSender
  - AsyncGetter
- **Benefits**
  - non-blocking operations
  - high-performance
  - multiple requests at the same time

Why Go Async?

- **Enhanced Performance**
  - More operations are handled concurrently
- **Better Resource Utilization**
  - More in the same timeframe
  - Utilize your multi-core
- **Improved Responsiveness**
  - Execute faster
  - Get answers faster

Async is simple

- Import asyncio
- Define function
- Use async class
- Call function

- Logout (!)

```python
import asyncio
from zabbix_utils import AsyncZabbixAPI, APIRequestError

# IDs of items for which the history should be cleared
ITEM_IDS = [70060, 70061, 70062]

async def main():

    # Create an instance of the AsyncZabbixAPI class
    api = AsyncZabbixAPI(url="192.168.0.1")

    # Authenticating with Zabbix API
    await api.login(user="Admin", password="zabbix")

    # Clear history for items with specified IDs
    try:
        await api.history.clear(*ITEM_IDS)
    except APIRequestError as e:
        print("An error occurred when attempting to delete items: ", e)
    else:
        # Logout to release the Zabbix API session
        await api.logout()

# Run the main coroutine
asyncio.run(main())

```

Summary & Wrap-Up

- **Understanding zabbix_utils**
  - Python library with 3 classes
    - ZabbixAPI
    - Sender
    - Getter
- **Growth and Community Impact**
  - More than **370k** in total downloads
  - Constant growth
  - Attention from community
- **Changes**
  - clusters and proxy groups for sender
  - SSL configuration and ssl_context
- **Async modules**

Where to start? How to Get?

ZABBIX

Introduction blog

Async blog-post

GitHub repository

Contact us

USA

Phone +1 877-4-ZABBIX
+1 877-4-922249 (Toll-free)

Email sales@zabbix.com

EUROPE

Phone +371 6778-4742

Email sales@zabbix.com

JAPAN

Phone +81 3-4405-7338

Email sales@zabbix.co.jp

CHINA

Phone +86 021-6978-6188

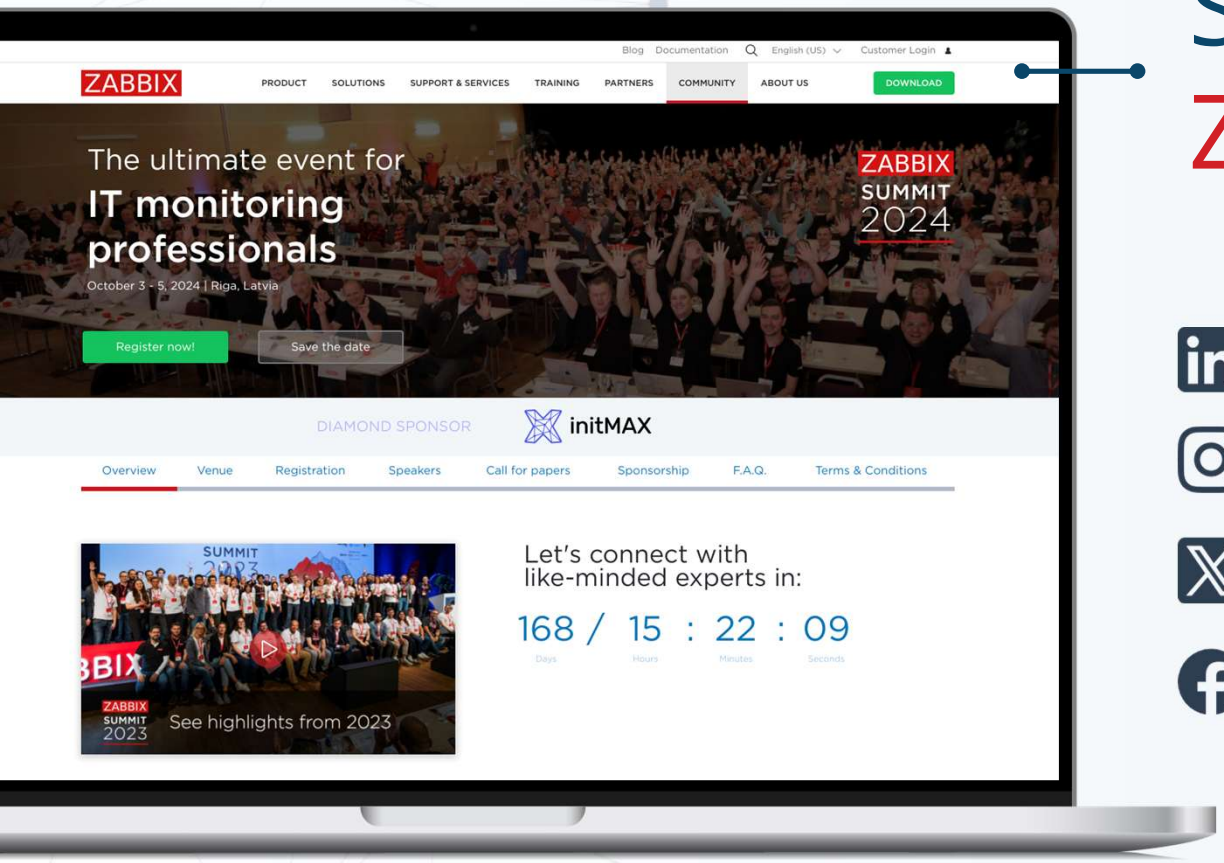
Email china@zabbix.co.jp

LATIN AMERICA

Phone	Argentina | Buenos Aires	+54 113989-4060
	Brazil | San Paulo	+55 11 4210-5104
	Chile |National	+56 44 890-9410
	Colombia | Bogota	+57 1 3819310
	Mexico | Mexico city	+52 55 8526-2606

Email sales.latam@zabbix.com

ZABBIX



Stay updated on Zabbix news:

- Zabbix
- zabbix_official
- @zabbix
- Zabbix

Aleksejs Abrosimovs

Intergration engineer