phpipam-pyclient Documentation

Release 0.1

Vinicius Arcanjo

Feb 22, 2022

Contents

CHAPTER 1

Introduction

phpipam-pyclient is a REST-client CLI tool to interface with PHPIpam REST API. phpipam-pyclient leverages python fire and requests under the hood, some high level functions have been implementend to allow the user to quickly query certain information about the devices on PHPIpam. In addition, you can use this library to build your Ansible inventory by filtering a field/column of the devices on PHPIpam.

1.1 Testing

Integration tests are implemented with pytest validating both Python2.7 and Python3.5 on a docker-based environment, in two stages:

- installation: validates a installation from strach with selenium.
- client-server API: validates this phpipam-pyclient with the phpipam REST API.

The following versions of PHPIpam are being tested on GitLab CI:

- 1.3.2
- 1.3.1
- 1.3

CHAPTER 2

Installation

2.1 via Github

1 - Git clone

```
git clone https://github.com/viniarck/phpipam-pyclient.git
cd phpipam-pyclient
```

2 - Install Python requirements dependencies, either via user install or virtualenv:

2.1 - pip user install:

```
pip install -e .
```

2.1 - or virtualenv:

```
virtualenv -p python3.6 .venv
source .venv/bin/activate
pip install -e .
```

CHAPTER 3

Configuration

In order to connect to PHPIpam REST API you have to edit the `phpipam_pyclient/config.json` file, which by default comes with the following configuration:

```
{
"base_url":"http://ipam/api",
"api_name":"testing",
"user":"admin",
"passwd":"my-secret-pw"
}
```

- `base_url`: This is the url of PHPIpam API `http(s)://<phpipam_server>/api`, make sure to adjust either http or https and the hostname of the PHPIpam server accordingly.
- `api_name`: The name of the API you have enabled on PHPIpam settings.
- `user`: username that will be authenticated on PHPIpam
- `passwd`: user's password

Optionally, if you don't want to specify another location for the *config.json* file, you can set the environment variable *PHPIPAM_PYCLIENT_CFG_FILE* which has higher precedence.

Note: When you enable API either choose ssl if you have https enabled or leave it as None for http. I haven't tested the crypto option.

CHAPTER 4

Usage

phpipam-client leverages python fire to implement the CLI, you can start by checking what options are available:

Note: Before you use this client, the PHPIpam server has to be up and running, since it's going to connect to it.

```
root@c0630eda943f:/app# phpipam-pyclient
Type:        PHPIpamClient
String form: <phpipam_pyclient.phpipam_pyclient.PHPIpamClient object at␣
↪0x7f8b49a44550>
Docstring:   PHPIPam Python API Client

Usage:       phpipam-pyclient -
             phpipam-pyclient - add-device
             phpipam-pyclient - ansible-inv-endpoint-field
             phpipam-pyclient - auth-session
             phpipam-pyclient - list-device-fields
             phpipam-pyclient - list-devices
             phpipam-pyclient - load-config
             phpipam-pyclient - version
```

Since I don't have any devices yet, let me start off by checking the arguments of the add-device function:

- input:

```
phpipam-pyclient - add-device -- --help
```

- output:

```
root@c0630eda943f:/app# phpipam-pyclient - add-device -- --help
Type:        method
String form: <bound method PHPIpamClient.add_device of <__main__.PHPIpamClient object␣
↪at 0x7fd016505828>>
File:        phpipam_pyclient.py
Line:        125
```

(continues on next page)

(continued from previous page)

```
Docstring:   Adds device to PHPIpam given a dictionary that represents a device
i.e., it should have these keys at least
'ip', 'hostname', 'description'

:device: dictionary that represents a device
:Returns: REST post status code

Usage:       phpipam-pyclient - add-device [DEVICE]
             phpipam-pyclient - add-device [--device DEVICE]
```

Let's add three devices on PHPIPam:

- input:

```
phpipam-pyclient add-device --device '{hostname:"server1",ip:"1.2.3.4",description:
↪"backend"}'
phpipam-pyclient add-device --device '{hostname:"server2",ip:"1.2.3.5",description:
↪"backend"}'
phpipam-pyclient add-device --device '{hostname:"server3",ip:"1.2.3.6",description:
↪"frontend"}'
```

- output

Note all REST calls returned 201 (OK) status code:

```
root@c0630eda943f:/app/phpipam_pyclient#   phpipam-pyclient add-device --device '
↪{hostname:"server1",ip:"1.2.3.4",description:"backend"}'
201
root@c0630eda943f:/app/phpipam_pyclient#   phpipam-pyclient add-device --device '
↪{hostname:"server2",ip:"1.2.3.5",description:"backend"}'
201
root@c0630eda943f:/app/phpipam_pyclient#   phpipam-pyclient add-device --device '
↪{hostname:"server3",ip:"1.2.3.6",description:"frontend"}'
201
root@c0630eda943f:/app/phpipam_pyclient#
```

Now, let's list all devices on PHPIPam:

- input:

```
phpipam-pyclient list-devices
```

- output:

```
root@c0630eda943f:/app/phpipam_pyclient# phpipam-pyclient list-devices
{"sections": "1;2", "snmp_v3_priv_protocol": "none", "snmp_queries": null, "hostname
↪": "server1", "snmp_port": "161", "rack_size": null, "id": "1", "location": null,
↪"snmp_v3_priv_pass": null, "description": "backend", "snmp_v3_auth_pass": null, "ip
↪": "1.2.3.4", "editDate": null, "snmp_v3_ctx_name": null, "snmp_timeout": "500",
↪"snmp_v3_auth_protocol": "none", "rack_start": null,"snmp_v3_ctx_engine_id": null,
↪"rack": null, "type": "0", "snmp_version": "0", "snmp_community": null, "snmp_v3_
↪sec_level": "none"}
{"sections": "1;2", "snmp_v3_priv_protocol": "none", "snmp_queries": null, "hostname
↪": "server2", "snmp_port": "161", "rack_size": null, "id": "2", "location": null,
↪"snmp_v3_priv_pass": null, "description": "backend", "snmp_v3_auth_pass": null, "ip
↪": "1.2.3.5", "editDate": null, "snmp_v3_ctx_name": null, "snmp_timeout": "500",
↪"snmp_v3_auth_protocol": "none", "rack_start": null,"snmp_v3_ctx_engine_id": null,
↪"rack": null, "type": "0", "snmp_version": "0", "snmp_community": null, "snmp_v3_
↪sec_level": "none"}
```

(continues on next page)

(continued from previous page)


```
{"sections": "1;2", "snmp_v3_priv_protocol": "none", "snmp_queries": null, "hostname
↪": "server3", "snmp_port": "161", "rack_size": null, "id": "3", "location": null,
↪"snmp_v3_priv_pass": null, "description": "frontend", "snmp_v3_auth_pass": null, "ip
↪": "1.2.3.6", "editDate": null, "snmp_v3_ctx_name": null, "snmp_timeout": "500",
↪"snmp_v3_auth_protocol": "none", "rack_start": null,"snmp_v3_ctx_engine_id": null,
↪"rack": null, "type": "0", "snmp_version": "0", "snmp_community": null, "snmp_v3_
↪sec_level": "none"}
```


Sweet! What if I wanted to export these devices as an Ansible inventory? It can group Ansible servers by their description, for example:

- input:

```
phpipam-pyclient ansible-inv-endpoint-field devices/ "description"
```

Note: Essentially, this command queries the devices/ endpoint and it'll group all hostnames according to their description, you could group by any other attribute if you wanted.

```
root@c0630eda943f:/app/phpipam_pyclient# phpipam-pyclient ansible-inv-endpoint-field␣
↪devices/ "description"
[frontend]
server3

[backend]
server1
server2
```

From this point forward, Ansible all the way to do whatever you need. But, what if you wanted to check all the other available fields what you could filter? If you had custom fields they would show up here too.

- input:

```
phpipam-pyclient list-device-fields
```

- output:

```
  root@c0630eda943f:/app/phpipam_pyclient# phpipam-pyclient list-device-fields
Type:         dict_keys
String form: dict_keys(['rack_size', 'snmp_v3_priv_pass', 'snmp_community', 'snmp_v3_
↪priv_protocol', 'sections', 'snmp_v3_ctx_name', 'snmp_v3_sec_level', 'editDate',
↪'rack_start', 'hostname', 'snmp_version', 'snmp_queries', 'snmp_v3_auth_pass',
↪'snmp_timeout', 'id', 'rack', 'description', 'location', 'snmp_v3_ctx_engine_id',
↪'ip', 'snmp_v3_auth_protocol', 'type', 'snmp_port'])
Length:       23

Usage:        phpipam_pyclient.py list-device-fields
              phpipam_pyclient.py list-device-fields isdisjoint
root@c0630eda943f:/app/phpipam_pyclient#
```

4.1 New Features

On version 1.0.0, released in Dec 2021, new filtering and Ansible grouping capabilities have been added:

Use and combine multiple filters (as a logic `and` operator) to filter based on any field that they have, for instance, let's say you wanted to filter if 'ip' fields contains the string '1.2.3' and also the 'description' is equal to 'backend'. These filter options are also available in the `ansible_inv_endpoint_field` command, here's it's being used on the `list_devices` command:

```
phpipam-pyclient list_devices --fields="['hostname', 'ip', 'description']" --filters=
↪"[{'type': 'contains', 'value': '1.2.3', 'field': 'ip'},{'type': 'eq', 'field':
↪'description', 'value': 'backend'}]"

{"hostname": "server1", "ip": "1.2.3.4", "description": "backend"}
{"hostname": "server2", "ip": "1.2.3.5", "description": "backend"}
```

If you need numerical comparisons, you can use the filter type as `ge, gt, le, gt` which respectively means greater than or equal, greater than, less than or equal, greater than. Another feature that was added was the support for adding Ansible default variables when generating the inventory, for instance, let's say you want to use these filters, group them by their `description`, but also, for any hosts that have the `description` as `frontend` you want to set the `ansible_port` as 2222, and `ansible_user` as `some_user`:

```
phpipam-pyclient ansible_inv_endpoint_field devices/ description --filters="[{'type':
↪'contains', 'value': '1', 'field': 'ip'}]" --ansible_kwargs="{'frontend': {'ansible_
↪port': 2222, ' ansible_user': 'some_user'}}"

[backend]
server1
server2

[frontend]
server3 ansible_port=2222 ansible_user=some_user
```

On top of that, if you also only want to include certain Ansible groups you can leverage the `--include_groups` option, notice that compared to the previous example, only the `frontend` group, that was grouped by their description is in the generated output:

```
phpipam-pyclient ansible_inv_endpoint_field devices/ description --filters="[{'type':
↪'contains', 'value': '1', 'field': 'ip'}]" --ansible_kwargs="{'frontend': {'ansible_
↪port': 2222, 'ansible_user': 'some_user'}}" --include_groups="['frontend']"

[frontend]
server3 ansible_port=2222 ansible_user=some_user
```