Worktory

Release 0.1.0

Renato Almeida de Oliveira

Oct 01, 2022

USING WORKTORY

Worktory is a python library created with the single purpose of simplifying the inventory management of network automation scripts.

As the network automation ecosystem grows, several connection plugins and parsers are available, and several times choosing a library or a connection plugin restricts all the devices to the same connection method.

Worktory tries to solve that problem giving the developer total flexibility for choosing the connector plugin and parsers for each device, at the same time that exposes a single interface for every plugin.

CHAPTER

ONE

INSTALLING WORKTORY

1.1 Installing

Worktory is available in PyPI, to install run:

```
$ pip install worktory
```

CHAPTER

TWO

INVENTORY CREATION

Currently, worktory accepts two input types for the inventory creation a *List[Dict]* and an *str* for the inventory file path, and for loading the inventory all you need is:

```
Inventory = InventoryManager("path")   # or
Inventory = InventoryManager(devices_list)
```

2.1 Device Object

The device object must have the following attributes:

- name
- hostname
- platform
- username
- password
- template_dir

Tip: Directory for custom textFSM templates

- select_parsers

Tip: Supports: 'genie', 'ntc', 'fsm', 'ALL', defaults to 'ALL'

- mode

Tip: Supports: "sync" or "async", defaults to "sync"

- parser

Tip: Defaults to "Default"

- connection_manager

Tip: Supports: ‘scrapli’, ‘unicon’, ‘netmiko’; defaults to ‘scrapli’

And custom attributes depending on which connector plugin, and parser you choose to use.

2.2 Unicon connector sample

```
devices = [{
    'name': 'sandbox-nxos',
    'hostname': 'sandbox-nxos-1.cisco.com',
    'platform': 'nxos',
    'username': 'admin',
    'password': 'Admin_1234!',
    'groups': ['CORE'],
    'connection_manager': 'unicon',
    'mode': 'sync',
    'transport': 'ssh',
}]
Inventory = InventoryManager(devices)
```

Optional attributes:

- ‘GRACEFUL_DISCONNECT_WAIT_SEC’, ‘POST_DISCONNECT_WAIT_SEC’, ‘conn_class’, ‘port’, ‘enable_password’,

2.3 Netmiko connector sample

```
devices = [{
    'name': 'sandbox-nxos',
    'hostname': 'sandbox-nxos-1.cisco.com',
    'platform': 'cisco_nxos',
    'username': 'admin',
    'password': 'Admin_1234!',
    'groups': ['CORE'],
    'connection_manager': 'netmiko',
    'mode': 'sync',
    'transport': 'ssh',
}]
Inventory = InventoryManager(devices)
```

Optional attributes:

- ‘port’, ‘verbose’, ‘global_delay_factor’, ‘global_cmd_verify’, ‘use_keys’, ‘key_file’, ‘pkey’, ‘passphrase’, ‘allow_agent’, ‘ssh_strict’, ‘system_host_keys’, ‘alt_host_keys’, ‘alt_key_file’, ‘ssh_config_file’, ‘conn_timeout’, ‘auth_timeout’, ‘banner_timeout’, ‘blocking_timeout’, ‘timeout’, ‘session_timeout’, ‘keepalive’, ‘default_enter’, ‘response_return’, ‘serial_settings’, ‘fast_cli’, ‘session_log’, ‘session_log_record_writes’, ‘session_log_file_mode’, ‘allow_auto_change’, ‘encoding’,

2.4 Scrapli sync connector sample

```
devices = [{
    'name': 'sandbox-nxos',
    'hostname': 'sandbox-nxos-1.cisco.com',
    'platform': 'cisco_nxos',
    'username': 'admin',
    'password': 'Admin_1234!',
    'groups': ['CORE'],
    'connection_manager': 'scrapli',
    'mode': 'sync',
}]
Inventory = InventoryManager(devices)
```

Optional attributes

- 'auth_private_key', 'auth_private_key_passphrase', 'auth_strict_key', 'auth_bypass', 'timeout_socket', 'transport', 'timeout_transport', 'timeout_ops', 'comms_prompt_pattern', 'comms_return_char', 'ssh_config_file', 'ssh_known_hosts_file', 'on_init', 'on_open', 'on_close', 'transport_options', 'channel_lock', 'channel_log', 'channel_log_mode', 'logging_uid', 'privilege_levels', 'default_desired_privilege_level', 'failed_when_contains',

2.5 Scrapli async connector sample

```
devices = [{
    'name': 'sandbox-nxos',
    'hostname': 'sandbox-nxos-1.cisco.com',
    'platform': 'cisco_nxos',
    'username': 'admin',
    'password': 'Admin_1234!',
    'groups': ['CORE'],
    'connection_manager': 'scrapli',
    'mode': 'async',
    'transport': 'asyncssh'
}]
Inventory = InventoryManager(devices)
```

Optional attributes

- 'auth_private_key', 'auth_private_key_passphrase', 'auth_strict_key', 'auth_bypass', 'timeout_socket', 'transport', 'timeout_transport', 'timeout_ops', 'comms_prompt_pattern', 'comms_return_char', 'ssh_config_file', 'ssh_known_hosts_file', 'on_init', 'on_open', 'on_close', 'transport_options', 'channel_lock', 'channel_log', 'channel_log_mode', 'logging_uid', 'privilege_levels', 'default_desired_privilege_level', 'failed_when_contains',

2.6 Using inventory file

The inventory file uses the yaml syntax, as bellow:

```
devices:
    'sandbox-nxos':
        'hostname': 'sandbox-nxos-1.cisco.com'
        'platform': 'cisco_nxos'
        'username': 'admin'
        'password': 'Admin_1234!'
        'groups':
        - 'CORE'
        'connection_manager': 'netmiko'
        'mode': 'sync'
        'transport': 'ssh'

    'sandbox-nxos-1':
        'hostname': 'sandbox-nxos-1.cisco.com'
        'platform': 'cisco_nxos'
        'username': 'admin'
        'password': 'Admin_1234!'
        'groups':
        - 'CORE'
        'connection_manager': 'scrapli'
        'mode': 'sync'

    'sandbox-nxos-2':
        'hostname': 'sandbox-nxos-1.cisco.com'
        'platform': 'nxos'
        'username': 'admin'
        'password': 'Admin_1234!'
        'groups':
        - 'CORE'
        'connection_manager': 'unicon'
        'mode': 'sync'
        'transport': 'ssh'
        'GRACEFUL_DISCONNECT_WAIT_SEC': 0
        'POST_DISCONNECT_WAIT_SEC': 0
```

For load the inventory file just:

```
Inventory = InventoryManager('inventory.yaml')
```

CHAPTER

THREE

FILTERING

The InventoryManager class implements a filter method that searches in every device for any attribute value and returns an iterator.

The returned iterator also implements the filter method which returns itself, that way the filter method can be concatenated to perform complex queries.

3.1 Sample Inventory

```
devices = [
            {
            'name': 'sandbox-nxos',
            'hostname': 'sandbox-nxos-1.cisco.com',
            'platform': 'cisco_nxos',
            'username': 'admin',
            'password': 'Admin_1234!',
            'groups': ['CORE'],
            'connection_manager': 'netmiko',
            'select_parsers' : 'genie',
            'mode': 'sync',
            'transport': 'ssh',
            },
            {
            'name': 'sandbox-nxos-1',
            'hostname': 'sandbox-nxos-1.cisco.com',
            'platform': 'cisco_nxos',
            'username': 'admin',
            'password': 'Admin_1234!',
            'groups': ['CORE'],
            'select_parsers' : 'ntc',
            'connection_manager': 'scrapli',
            'mode': 'async',
            'transport': 'asyncssh'
            },
            {
            'name': 'sandbox-nxos-2',
            'hostname': 'sandbox-nxos-1.cisco.com',
            'platform': 'nxos',
            'username': 'admin',
            'password': 'Admin_1234!',
```

(continues on next page)

(continued from previous page)

```
            'groups': ['EDGE'],
            'connection_manager': 'unicon',
            'mode': 'sync',
            'transport': 'ssh',
            'GRACEFUL_DISCONNECT_WAIT_SEC': 0,
            'POST_DISCONNECT_WAIT_SEC': 0,
            }
        ]
```

3.2 Filtering by mode

```
>>> from worktory import InventoryManager
>>> inventory = InventoryManager(devices)
>>> print([device.name for device in inventory.filter(mode='async')])
['sandbox-nxos-1']
```

3.3 Filtering by groups

```
>>> from worktory import InventoryManager
>>> inventory = InventoryManager(devices)
>>> print([device.name for device in inventory.filter(groups='EDGE')])
['sandbox-nxos-2']
```

3.4 Filtering by parsers

```
>>> from worktory import InventoryManager
>>> inventory = InventoryManager(devices)
>>> print([device.name for device in inventory.filter(select_parsers='ntc')])
['sandbox-nxos-2', 'sandbox-nxos-1']
```

Tip: If select_parsers attribute isn't set worktory default behavior is to use all available parsers

3.5 Filtering by parser and group

```
>>> from worktory import InventoryManager
>>> inventory = InventoryManager(devices)
>>> print([device.name for device in inventory.filter(select_parsers='ntc',
...                                                     groups='CORE',
...                                                     filter_mode="AND")])
['sandbox-nxos-1']
```

3.6 Concatenating filters

```
>>> from worktory import InventoryManager
>>> inventory = InventoryManager(devices)
>>> print([device.name for device in inventory.filter(select_parsers='ntc').
↪filter(groups='CORE')])
['sandbox-nxos-1']
```

CHAPTER

FOUR

USING WORKTORY

4.1 Sample Inventory

```
devices = [
            {
            'name': 'sandbox-iosxr-1',
            'hostname': 'sandbox-iosxr-1.cisco.com',
            'platform': 'cisco_iosxr',
            'username': 'admin',
            'password': 'C1sco12345',
            'groups': ['CORE'],
            'connection_manager': 'scrapli',
            'select_parsers' : 'genie',
            'mode': 'async',
            'transport': 'asyncssh',
            },
            {
            'name': 'sandbox-nxos-1',
            'hostname': 'sandbox-nxos-1.cisco.com',
            'platform': 'cisco_nxos',
            'username': 'admin',
            'password': 'Admin_1234!',
            'groups': ['CORE'],
            'select_parsers' : 'ntc',
            'connection_manager': 'scrapli',
            'mode': 'async',
            'transport': 'asyncssh'
            },
            {
            'name': 'sandbox-nxos-2',
            'hostname': 'sandbox-nxos-1.cisco.com',
            'platform': 'nxos',
            'username': 'admin',
            'password': 'Admin_1234!',
            'groups': ['EDGE'],
            'connection_manager': 'unicon',
            'mode': 'sync',
            'transport': 'ssh',
            'GRACEFUL_DISCONNECT_WAIT_SEC': 0,
            'POST_DISCONNECT_WAIT_SEC': 0,
```

(continues on next page)

(continued from previous page)

```
            },
            {
            'name': 'sandbox-iosxr-2',
            'hostname': 'sandbox-iosxr-1.cisco.com',
            'platform': 'cisco_iosxr',
            'username': 'admin',
            'password': 'C1sco12345',
            'groups': ['CORE'],
            'connection_manager': 'scrapli',
            'select_parsers' : 'genie',
            'mode': 'sync',
            },
        ]
```

4.2 Collecting Running config from async devices

```
from worktory import InventoryManager
import asyncio
inventory = InventoryManager(devices)

device_configs = {}
async def get_config(device):
    await device.connect()
    config = await device.execute("show running-config")
    device_configs[device.name] = config
    await device.disconnect()

async def async_main():
    coros = [get_config(device) for device in inventory.filter(mode='async')]
    await asyncio.gather(*coros)

loop = asyncio.get_event_loop()
loop.run_until_complete(async_main())
```

4.3 Collecting Running config from sync devices

```
from worktory import InventoryManager
from multiprocessing import Pool
inventory = InventoryManager(devices)

def get_config(device_name):
    inventory = InventoryManager(devices)
    device = inventory.devices[device_name]
    device.connect()
    config = device.execute("show running-config")
    device.disconnect()
    return ( device.name , config )
```

(continues on next page)

(continued from previous page)

```
def main():
    devs = [device.name for device in inventory.filter(mode='sync')]
    with Pool(2) as p:
        return p.map(get_config, devs)


output = main()
```

CHAPTER FIVE

PARSING

By default worktory tries to parse the device output in all available parsers, ie, *ntc-templates*, *genie parses*, and custom *textFSM*.

To use custom textFSM devices, create the following directive structure in your working directory.

```
.
├── script.py
├── inventory.yml
├── parsers
├    ├── platform
├    ├    ├── command.textfsm
├    ├── comware
├    ├    ├── display_interfaces.textfsm
```

Tip: Worktory replace "spaces" by "_" when looking for the appropriated parser

5.1 Code example

5.1.1 sample inventory

```
devices = [
            {
            'name': 'sandbox-iosxr-1',
            'hostname': 'sandbox-iosxr-1.cisco.com',
            'platform': 'cisco_iosxr',
            'username': 'admin',
            'password': 'C1sco12345',
            'groups': ['CORE'],
            'connection_manager': 'scrapli',
            'select_parsers' : 'genie',
            'mode': 'async',
            'transport': 'asyncssh',
            },
            {
            'name': 'sandbox-nxos-1',
            'hostname': 'sandbox-nxos-1.cisco.com',
            'platform': 'cisco_nxos',
```

(continues on next page)

(continued from previous page)

```
            'username': 'admin',
            'password': 'Admin_1234!',
            'groups': ['CORE'],
            'select_parsers' : 'ntc',
            'connection_manager': 'scrapli',
            'mode': 'async',
            'transport': 'asyncssh'
            },
            {
            'name': 'sandbox-nxos-2',
            'hostname': 'sandbox-nxos-1.cisco.com',
            'platform': 'cisco_nxos',
            'username': 'admin',
            'password': 'Admin_1234!',
            'groups': ['EDGE'],
            'connection_manager': 'scrapli',
            'mode': 'sync',
            },
            {
            'name': 'sandbox-iosxr-2',
            'hostname': 'sandbox-iosxr-1.cisco.com',
            'platform': 'cisco_iosxr',
            'username': 'admin',
            'password': 'C1sco12345',
            'groups': ['CORE'],
            'connection_manager': 'scrapli',
            'select_parsers' : 'genie',
            'mode': 'sync',
            },
        ]
```

5.1.2 Parsing show interfaces

```
>>> from worktory import InventoryManager
>>> inventory = InventoryManager(devices)
>>> device = inventory.devices['sandbox-nxos-2']
>>> device.connect()
>>> output = device.parse("show version")
>>> device.disconnect()
>>> print(output)
{ 'fsm': {
     'fail': "[Errno 2] No such file or directory: '/home/renato/Worktory/parsers/cisco_
↪nxos/show_version.textfsm'"},
 'ntc': {
     'result': [{'uptime': '0 day(s), 6 hour(s), 59 minute(s), 22 second(s)', 'last_
↪reboot_reason': 'Unknown', 'os': '9.3(3)', 'boot_image': 'bootflash:///nxos.9.3.3.bin',
↪ 'platform': 'C9300v', 'hostname': 'NXOS-Always-On', 'serial': '9N3KD63KWT0'}]},
 'genie': {
     'result': {'platform': {'name': 'Nexus', 'os': 'NX-OS', 'software': {'system_version
↪': '9.3(3)', 'system_image_file': 'bootflash:///nxos.9.3.3.bin', 'system_compile_time
```

(continues on next page)

(continued from previous page)

```
↪': '12/22/2019 2:00:00 [12/22/2019 14:00:37]'}, 'hardware': {'model': 'Nexus9000 C9300v
↪', 'chassis': 'Nexus9000 C9300v', 'slots': 'None', 'rp': 'None', 'cpu': 'Intel(R)␣
↪Xeon(R) Gold 6148 CPU @ 2.40GHz', 'memory': '16408988 kB', 'processor_board_id':
↪'9N3KD63KWT0', 'device_name': 'NXOS-Always-On', 'bootflash': '4287040 kB'}, 'kernel_
↪uptime': {'days': 0, 'hours': 6, 'minutes': 59, 'seconds': 22}, 'reason': 'Unknown'}}}}
```

CHAPTER

SIX

WORKTORY

6.1 worktory package

6.1.1 Subpackages

worktory.connection package

Subpackages

worktory.connection.wrappers package

Submodules

worktory.connection.wrappers.netmiko_wrapper module

worktory.connection.wrappers.scrapli_wrapper module

worktory.connection.wrappers.unicon_wrapper module

Module contents

Submodules

worktory.connection.base_wrapper module

worktory.connection.fabric module

Module contents

worktory.device package

Submodules

worktory.device.device module

Module contents

worktory.inventory package

Submodules

worktory.inventory.inventory module

Module contents

worktory.parsers package

Subpackages

worktory.parsers.wrappers package

Submodules

worktory.parsers.wrappers.default module

Module contents

Submodules

worktory.parsers.base_parser module

exception `worktory.parsers.base_parser.`**`MethodNotImplemented`**

Bases: `Exception`

Raises when wrapper doesn't implement

class `worktory.parsers.base_parser.`**`base_parser`**(*device: Device*)

Bases: `object`

`configure`(*device: Device*) → Dict[str, Callable]

`required_interfaces: List[str] = ['parse']`

worktory.parsers.fabric module

Module contents

6.1.2 Module contents

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX