

Ansible로 네트워크 장비 정보 수집 해보기 - IOSXE

DevNet Team Technical Solutions Specialist

김혜영 프로



Hands-on 순서

- ios_facts & ios_command
- Pyats_parse_command
- 미션



Hands on 시작 전 세팅

[입력 필요]

- 1) `cd ~/DevNet_Korea/02_IOSXE/`
- 2) `ls`
- 3) `ssh admin@192.168.14.14`
- 4) 비밀번호 : Cisco123!@#
- 5) `exit`

```
devnet@devnet-virtual-machine:~/DevNet_Korea$ cd ~/DevNet_Korea/02_IOSXE/
devnet@devnet-virtual-machine:~/DevNet_Korea/02_IOSXE$ ls
02_count_up_interface.yaml  02_mission_answer.yaml  host_vars
02_get_switch_info.yaml    02_mission.yaml        README
02_ios_command.yaml        ansible.cfg              requirements.yaml
02_ios_facts.yaml          hosts
```

Hands on 구조 설명 - host

```
devnet@devnet-virtual-machine:~/DevNet_Korea/02_IOSXE$ ls
02_count_up_interface.yaml  02_mission_answer.yaml  host_vars
02_get_switch_info.yaml    02_mission.yaml        README
02_ios_command.yaml        ansible.cfg              requirements.yaml
02_ios_facts.yaml          hosts
```

```
devnet@devnet-virtual-machine:~/DevNet_Korea/02_IOSXE$ cat hosts
[all:vars]
ansible_connection = ansible.netcommon.network_cli
ansible_user = admin
ansible_password = Cisco123!@#

[switches]
cat9k-1
```

host_vars 디렉토리

cat9k-1.yaml

Hands on 구조 설명 - host

```
devnet@devnet-virtual-machine:~/DevNet_Korea/02_IOSXE$ ls
02_count_up_interface.yaml  02_mission_answer.yaml  host_vars
02_get_switch_info.yaml    02_mission.yaml        README
02_ios_command.yaml        ansible.cfg              requirements.yaml
02_ios_facts.yaml          hosts
```

```
devnet@devnet-virtual-machine:~/DevNet_Korea/02_IOSXE$ cat hosts
[all:vars]
ansible_connection = ansible.netcommon.network_cli
ansible_user = admin
ansible_password = Cisco123!@#

[switches]
cat9k-1devnet@devnet-virtual-machine:~/DevNet_Korea/02_IOSXE$
```

```
devnet@devnet-virtual-machine:~/DevNet_Korea/02_IOSXE$ cat host_vars/cat9k-1.yaml
ansible_host: 192.168.14.14
ansible_network_os: iosdevnet@devnet-virtual-machine:~/DevNet_Korea/02_IOSXE$
```

Hands on 구조 설명 - playbook

```
devnet@devnet-virtual-machine:~/DevNet_Korea/02_IOSXE$ ls
02_count_up_interface.yaml  02_mission_answer.yaml  host_vars
02_get_switch_info.yaml    02_mission.yaml         README
02_ios_command.yaml        ansible.cfg              requirements.yaml
02_ios_facts.yaml          hosts
```

IOS-XE 장비에서 facts 정보 가져오기

02_ios_facts.yaml

- 목표:
- ios-xe 장비의 호스트네임, 시리얼 넘버, 인터페이스 정보 가져오기
- 방법:
- ios_facts 모듈을 사용하여 ios-xe 장비의 facts 정보를 가져오기
- 실행:

```
devnet@devnet-virtual-machine:~/DevNet_Korea/02_IOSXE$ ansible-playbook 02_ios_facts.yaml
```

IOS-XE 장비에서 facts 정보 가져오기

ios_facts 모듈

```
- hosts: switches
  gather_facts: no

tasks:
  - name: 1. ios facts 데이터
    cisco.ios.ios_facts:
      gather_subset: all

  - name: 2. 호스트네임 & 시리얼 번호
    debug:
      msg: "Hostname is {{ ansible_net_serialnum }}"

  - name: 3. 인터페이스 정보
    debug:
      var: ansible_net_interfaces
```

ios_facts

ios/ios-xe 장비에서 fact를 가져올 때 사용하는 모듈

자료 :

https://docs.ansible.com/ansible/latest/collections/cisco/ios/ios_facts_module.html

Return Values

Common return values are documented [here](#), the following are the fields unique to this module:

| Key | Description |
|--|--|
| <code>ansible_net_all_ipv4_addresses</code> <small>list / elements=string</small> | All IPv4 addresses configured on the device Returned: when interfaces is configured |
| <code>ansible_net_all_ipv6_addresses</code> <small>list / elements=string</small> | All IPv6 addresses configured on the device Returned: when interfaces is configured |
| <code>ansible_net_api</code> <small>string</small> | The name of the transport Returned: always |
| <code>ansible_net_config</code> <small>string</small> | The current active config from the device Returned: when config is configured |
| <code>ansible_net_cpu_utilization</code> <small>dictionary</small> | The current CPU utilization of the device Returned: when hardware is configured |
| <code>ansible_net_filesystems</code> <small>list / elements=string</small> | All file system names available on the device Returned: when hardware is configured |

IOS-XE 장비에서 facts 정보 가져오기

los_facts 모듈

```
- hosts: switches
  gather_facts: no

tasks:
  - name: 1. ios facts 데오
    cisco.ios.ios_facts:
      gather_subset: all

  - name: 2. 호스트네임 &
    serial number)
    debug:
      msg: "Hostname is {{ ansible_net_hostname }} and the Serial Number is {{
      ansible_net_serialnum }}"

  - name: 3. 인터페이스 정보 보여주기 (Print out interface Information)
    debug:
      var: ansible_net_interfaces
```

```
devnet@devnet-virtual-machine:~/DevNet_Korea/02_IOSXE$ ansible-playbook 02_ios_facts.yaml

PLAY [switches] *****

TASK [1. ios facts 데이터 가져오기 (retrieve ios facts)] *****
[WARNING]: ansible-pylibssh not installed, falling back to paramiko
ok: [cat9k-1]

TASK [2. 호스트네임 & 시리얼 넘버 보여주기 (Print out the hostname and serial number)] ****
ok: [cat9k-1] => {
  "msg": "Hostname is cat9kv1 and the Serial Number is 9M2ST6PVKOA"
}
```

IOS-XE 장비에서 facts 정보 가져오기

los_facts 모듈

```
- hosts: switches
  gather_facts: no

tasks:
  - name: 1. ios facts 데이터 가져오기 (retrieve ios facts)
    cisco.ios.ios_facts:
      gather_subset: all

  - name: 2. 호스트네임 & 시리얼 넘버 보여주기 (Print out host name & serial number)
    debug:
      msg: "Hostname is {{ ansible_net_hostname }} and serial number is {{ ansible_net_serialnum }}"

  - name: 3. 인터페이스 정보 보여주기 (Print out interface information)
    debug:
      var: ansible_net_interfaces
```

```
TASK [3. 인터페이스 정보 보여주기 (Print out interface Information)] *****
ok: [cat9k-1] => {
  "ansible_net_interfaces": {
    "GigabitEthernet0/0": {
      "bandwidth": 1000000,
      "description": null,
      "duplex": "Full",
      "ipv4": [
        {
          "address": "192.168.14.14",
          "subnet": "24"
        }
      ],
      "lineprotocol": "up",
      "macaddress": "5254.001b.293f",
      "mediatype": "RJ45",
      "mtu": 1500,
      "operstatus": "up",
      "type": "RP management port"
    },
    "GigabitEthernet1/0/1": {
      "bandwidth": 1000000,
      "description": null,
      "duplex": null,
      "ipv4": [],
      "lineprotocol": "up",

```

Facts에 없는 값?

```
cat9kv1#show version
Cisco IOS XE Software, Version 17.10.01prd7
Cisco IOS Software [Dublin], Catalyst L3 Switch Software (CAT9K_IOSXE), Version 17.10.1prd7, RELEASE SOFTWARE (fc1)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2022 by Cisco Systems, Inc.
Compiled Wed 21-Sep-22 22:33 by mcpre
```

```
Cisco IOS-XE software, Copyright (c) 2005-2022 by cisco Systems, Inc.
All rights reserved. Certain components of Cisco IOS-XE software are
licensed under the GNU General Public License ("GPL") Version 2.0. The
software code licensed under GPL Version 2.0 is free software that comes
with ABSOLUTELY NO WARRANTY. You can redistribute and/or modify such
GPL code under the terms of GPL Version 2.0. For more details, see the
documentation or "License Notice" file accompanying the IOS-XE software,
or the applicable URL provided on the flyer accompanying the IOS-XE
software.
```

```
ROM: IOS-XE ROMMON
```

```
BOOTLDR:
```

```
cat9kv1 uptime is 4 days, 19 hours, 31 minutes
```

```
Uptime for this control processor is 4 days, 19 hours, 34 minutes
```

```
System returned to ROM by unknown reload cause - reason ptr 0xF, PC 0x0, address 0x0
```

```
System image file is "bootflash:packages.conf"
```

```
Last reload reason: unknown reload cause - reason ptr 0xF, PC 0x0, address 0x0
```

ios_command 모듈

커맨드 결과를 가져오는 모듈

플레이북

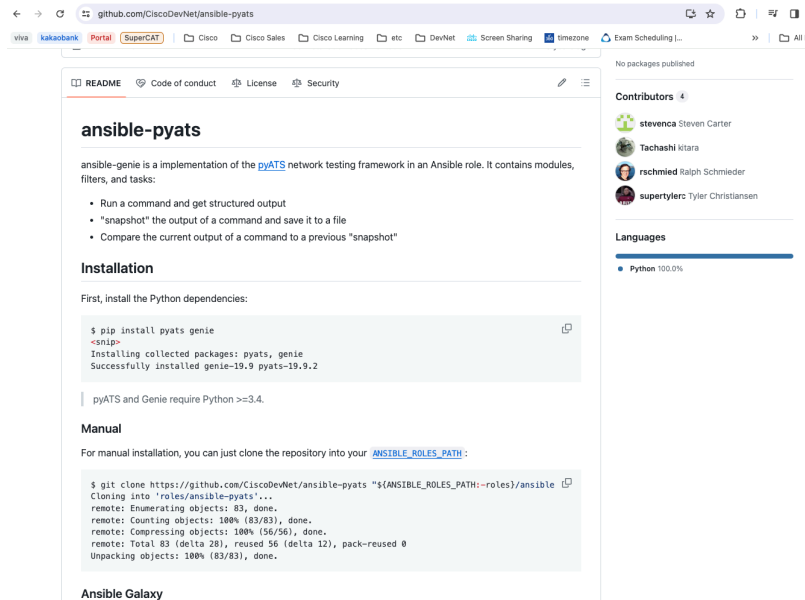
```
1 - name: Sample IOS XE playbook to get the command output
2   hosts: switches
3   connection: network_cli
4   gather_facts: no
5
6   tasks:
7     - name: Get the output of the command 'show version'
8       cisco.ios.ios_command:
9         commands: show version
10        register: command_output
11
12    - name: Print out the command output
13      debug:
14        var: command_output
15
```

실행 결과

```
"stdout_lines": [
  [
    "Cisco IOS XE Software, Version S2C",
    "Cisco IOS Software [Dublin], Catalyst L3 Switch Software (CAT9K_IOSXE), Experim",
    "cpre/s2c-build-ws 101]",
    "Copyright (c) 1986-2022 by Cisco Systems, Inc.",
    "Compiled Thu 08-Sep-22 09:57 by mcpre",
    "",
    "Cisco IOS-XE software, Copyright (c) 2005-2022 by cisco Systems, Inc.",
    "All rights reserved. Certain components of Cisco IOS-XE software are",
    "licensed under the GNU General Public License (\"GPL\") Version 2.0. The",
    "software code licensed under GPL Version 2.0 is free software that comes",
    "with ABSOLUTELY NO WARRANTY. You can redistribute and/or modify such",
    "GPL code under the terms of GPL Version 2.0. For more details, see the",
    "documentation or \"License Notice\" file accompanying the IOS-XE software,",
    "or the applicable URL provided on the flyer accompanying the IOS-XE",
    "software.",
    "",
    "ROM: IOS-XE ROMMON",
    "Cat9kv-01 uptime is 3 hours, 8 minutes",
    "uptime for this control processor is 3 hours, 10 minutes",
    "System returned to ROM by Reload Command",
    "System image file is \"bootflash:packages.conf\"",
    "Last reload reason: Reload Command",
    "",
    "This product contains cryptographic features and is subject to United",
    "States and local country laws governing import, export, transfer and",
    "use. Delivery of Cisco cryptographic products does not imply",
    "third-party authority to import, export, distribute or use encryption.",
    "Importers, exporters, distributors and users are responsible for",
    "compliance with U.S. and local country laws. By using this product you",
    "agree to comply with applicable laws and regulations. If you are unable",
    "to comply with U.S. and local laws, return this product immediately.",
    "",
    "A summary of U.S. laws governing Cisco cryptographic products may be found at:",
    "http://www.cisco.com/wwl/export/crypto/tool/stqrg.html",
    "If you require further assistance please contact us by sending email to",
    "export@cisco.com.",
    "",
    "Technology Package License Information: ",
    "",
    "Technology-package",
    "Current      Type      Next reboot",
    "network-essentials  \tSmart License  \t network-essentials ",
    "None          \tSubscription Smart License  \t None",
    "AIR License Level: AIR DNA Advantage"
  ]
]
```

Ansible-pyats

Ansible에서 사용할 수 있는 pyats parser



github.com/CiscoDevNet/ansible-pyats

ansibler-**pyats**

ansible-genie is an implementation of the [pyATS](#) network testing framework in an Ansible role. It contains modules, filters, and tasks:

- Run a command and get structured output
- "snapshot" the output of a command and save it to a file
- Compare the current output of a command to a previous "snapshot"

Installation

First, install the Python dependencies:

```
$ pip install pyats genie
<snip>
Installing collected packages: pyats, genie
Successfully installed genie-19.9 pyats-19.9.2
```

pyATS and Genie require Python >=3.4.

Manual

For manual installation, you can just clone the repository into your [ANSIBLE_ROLES_PATH](#):

```
$ git clone https://github.com/CiscoDevNet/ansible-pyats "${ANSIBLE_ROLES_PATH:-roles}/ansible
Cloning into 'roles/ansible-pyats'...
remote: Enumerating objects: 83, done.
remote: Counting objects: 100% (83/83), done.
remote: Compressing objects: 100% (56/56), done.
remote: Total 83 (delta 28), reused 56 (delta 12), pack-reused 0
Unpacking objects: 100% (83/83), done.
```

Contributors

- stevenca Steven Carter
- Tachashi Kitara
- rschmied Ralph Schmieder
- supertylerz Tyler Christiansen

Languages

- Python 100.0%

<https://github.com/CiscoDevNet/ansible-pyats>

Ansible-pyats

```
cat9kv1#show version
Cisco IOS XE Software, Version 17.10.01prd7
Cisco IOS Software [Dublin], Catalyst L3 Switch Software (CAT9K_IOSXE), Version 17.10.1prd7,
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2022 by Cisco Systems, Inc.
Compiled Wed 21-Sep-22 22:33 by mcpre
```

```
Cisco IOS-XE software, Copyright (c) 2005-2022 by cisco Systems, Inc.
All rights reserved. Certain components of Cisco IOS-XE software are
licensed under the GNU General Public License ("GPL") Version 2.0. The
software code licensed under GPL Version 2.0 is free software that comes
with ABSOLUTELY NO WARRANTY. You can redistribute and/or modify such
GPL code under the terms of GPL Version 2.0. For more details, see the
documentation or "License Notice" file accompanying the IOS-XE software,
or the applicable URL provided on the flyer accompanying the IOS-XE
software.
```

```
ROM: IOS-XE ROMMON
BOOTLDR:
cat9kv1 uptime is 4 days, 19 hours, 31 minutes
Uptime for this control processor is 4 days, 19 hours, 34 minutes
System returned to ROM by unknown reload cause - reason ptr 0xF, PC 0x0, address 0x0
System image file is "bootflash:packages.conf"
Last reload reason: unknown reload cause - reason ptr 0xF, PC 0x0, address 0x0
```

```
"structured": {
  "version": {
    "air_license_level": "AIR DNA Advantage",
    "chassis": "C9KV-UADP-8P",
    "chassis_sn": "9M2ST6PVK0A",
    "compiled_by": "mcpre",
    "compiled_date": "Wed 21-Sep-22 22:33",
    "copyright_years": "1986-2022",
    "curr_config_register": "0x2102",
    "disks": {
      "bootflash.": {
        "disk_size": "5236224",
        "type_of_disk": "virtual hard disk"
      }
    },
    "hostname": "cat9kv1",
    "image_id": "CAT9K_IOSXE",
    "image_type": "production image",
    "label": "RELEASE SOFTWARE (fc1)",
    "last_reload_reason": "unknown reload cause - reason ptr 0xF, PC 0x0, address 0x0",
    "license_package": {
      "None": {
        "license_level": "None",
        "license_type": "Subscription Smart License",
        "next_reload_license_level": "None"
      },
      "network-essentials": {
        "license_level": "network-essentials",
        "license_type": "Smart License",
        "next_reload_license_level": "network-essentials"
      }
    },
    "location": "Dublin",
    "main_mem": "759406",
    "mem_size": {
      "non-volatile configuration": "32768",
      "physical": "18874368"
    }
  }
}
```

Ansible-pyats 사용 방법

1. 설치 매뉴얼에 따라 설치를 한다

Installation

First, install the Python dependencies:

```
$ pip install pyats genie
<snip>
Installing collected packages: pyats, genie
Successfully installed genie-19.9 pyats-19.9.2
```

pyATS and Genie require Python >=3.4.

Manual

For manual installation, you can just clone the repository into your

[ANSIBLE_ROLES_PATH](#):

```
$ git clone https://github.com/CiscoDevNet/ansible-pyats "${ANSIBLE_ROLES_PATH}"
Cloning into 'roles/ansible-pyats'...
remote: Enumerating objects: 83, done.
remote: Counting objects: 100% (83/83), done.
remote: Compressing objects: 100% (56/56), done.
remote: Total 83 (delta 28), reused 56 (delta 12), pack-reused
Unpacking objects: 100% (83/83), done.
```

Ansible Galaxy

If you are using [Ansible Galaxy](#), you can use this role by adding the following to your `requirements.yml`:

```
- src: https://github.com/CiscoDevNet/ansible-pyats
  scm: git
  name: ansible-pyats
```

Next, install your Galaxy dependencies:

```
$ ansible-galaxy install -r requirements.yml -p "${ANSIBLE_ROLES_PATH}"
```

2. 설치된 role 을 이용해서 playbook을 만든다

```
- hosts: switches
  gather_facts: no
  roles:
    - ../roles/ansible-pyats
  tasks:
    - name: 1. show version 커맨드 결과값 가져오기 (Get the o
      pyats_parse_command:
```

PyATS Parser List

파싱 가능한 커맨드 리스트

Genie

Parsers List

all

XE ping mpls ip {addr} {mask} rep

XE ping mpls traffic-eng tunnel {tu

XE ping {addr}

XE ping {addr} source {source} rep

XE sh lsp locator-table {locator_ta

XE map-cache reverse-address-res

XE show aaa cache group {server

XE show aaa dead-criteria radius

XE show aaa dead-criteria radius {

XE show aaa memory

XE show aaa servers

XE show aaa user all

XE show access-lists {acl}

XE show access-session brief

ALL

IOS

IOSXE

IOSXR

NXOS

ASA

LINUX

JUNOS

SROS

BIGIP

VIPTTELA

APIC

DNAC

IRONWARE

AIREOS

CHEETAH

GAIA

GENERIC

COMWARE

XE ping mpls pseudowire {addr} {vc_id}

XE ping vrf {vrf} {addr}

XE ping {addr} Extended-data {extended_data}

XE sh lsp locator-table {locator_table} instance-id {instance_id} ethernet

XE map-cache reverse-address-resolution

XE show aaa cache group {server_grp} all

XE show aaa common-criteria policy name {policy_name}

XE show aaa dead-criteria radius {server_ip}

XE show aaa fqdn all

XE show aaa method-lists {type}

XE show aaa sessions

XE show access-lists

XE show access-session

XE show access-session info

<https://pubhub.devnetcloud.com/media/genie-feature-browser/docs/#/parsers>

[입력 필요]

1) ansible-playbook 02_get_switch_info.yaml

PyATS 예제 1) 스위치 정보 가져오기

목표

스위치의 Hostname, Version, Serial Number, Uptime, cpu 정보 가져오기

방법

Ansible-pyats를 사용하여 파싱 된 show version & show processes cpu 커맨드의 결과 값을 사용하기

실행 방법

```
devnet@devnet-virtual-machine:~/DevNet_Korea/02_IOSXE$ ansible-playbook 02_get_switch_info.yaml
```

PyATS 예제 1) 스위치 정보 가져오기

과정 - playbook의 Task

```
tasks:
- name: 1. show version 커맨드 결과값 가져오기 (Get the output of the command "show version")
  pyats_parse_command:
    command: show version
    register: version_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)
  debug:
    msg:
      - "{{version_output.structured}}"

- name: 3. 스위치 정보 보여주기 - Hostname, Version, Serial Number, Uptime (Display Switch Information - Hostname, Version, Serial Number, Uptime)
  debug:
    msg:
      - "Hostname: {{ version_output.structured.version.hostname }}"
      - "Version: {{ version_output.structured.version.version }}"
      - "Serial Number: {{ version_output.structured.version.chassis_sn }}"
      - "Uptime: {{ version_output.structured.version.uptime }}"

- name: 4. show processes cpu 커맨드 결과값 가져오기 (Get the output of the command "show processes cpu")
  pyats_parse_command:
    command: show processes cpu
    register: cpu_output

- name: 5. CPU 5분 평균 사용률 보여주기 (Print out the average cpu usage(5 minutes))
  debug:
    msg:
      - "CPU: {{ cpu_output.structured.five_min_cpu }}%"
```

PyATS 예제 1) 스위치 정보 가져오기

과정 - playbook의 Task

```
tasks:
- name: 1. show version 커맨드 결과값 가져오기 (Get the output of the command)
  pyats_parse_command:
    command: show version
    register: version_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)
  debug:
    msg:
      - "{{version_output.structured}}"

- name: 3. 스위치 정보 보여주기 - Hostname, Version, Serial Number, Uptime (D)
  debug:
    msg:
      - "Hostname: {{ version_output.structured.version.hostname }}"
      - "Version: {{ version_output.structured.version.version }}"
      - "Serial Number: {{ version_output.structured.version.chassis_sn }}"
      - "Uptime: {{ version_output.structured.version.uptime }}"

- name: 4. show processes cpu 커맨드 결과값 가져오기 (Get the output of the command)
  pyats_parse_command:
    command: show processes cpu
    register: cpu_output

- name: 5. CPU 5분 평균 사용률 보여주기 (Print out the average cpu usage(5 min))
  debug:
    msg:
      - "CPU: {{ cpu_output.structured.five_min_cpu }}"
```

```
TASK [2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)] *****
ok: [cat9k-1] => {
  "msg": [
    {
      "version": {
        "air_license_level": "AIR DNA Advantage",
        "chassis": "C9KV-UADP-8P",
        "chassis_sn": "9M2ST6PVK0A",
        "compiled_by": "mcpres",
        "compiled_date": "Wed 21-Sep-22 22:33",
        "copyright_years": "1986-2022",
        "curr_config_register": "0x2102",
        "disks": {
          "bootflash:": {
            "disk_size": "5236224",
            "type_of_disk": "virtual hard disk"
          }
        },
        "hostname": "cat9kv1",
        "image_id": "CAT9K_IOSXE",
        "image_type": "production image",
        "label": "RELEASE SOFTWARE (fc1)",
        "last_reload_reason": "unknown reload cause - reason ptr 0xF, PC 0x0, address 0x0",
        "license_package": {
          "None": {
            "license_level": "None",
            "license_type": "Subscription Smart License",
            "next_reload_license_level": "None"
          },
          "network-essentials": {
            "license_level": "network-essentials",
            "license_type": "Smart License",
            "next_reload_license_level": "network-essentials"
          }
        },
        "location": "Dublin",
```

PyATS 예제 1) 스위치 정보 가져오기

과정 - playbook의 Task

```
tasks:
- name: 1. show version 커맨드 결과값 가져오기 (Get the output of the command)
  pyats_parse_command:
    command: show version
    register: version_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)
  debug:
    msg:
      - "{{version_output.structured}}"

- name: 3. 스위치 정보 보여주기 - Hostname, Version, Serial Number, Uptime (D
  debug:
    msg:
      - "Hostname: {{ version_output.structured.version.hostname }}"
      - "Version: {{ version_output.structured.version.version }}"
      - "Serial Number: {{ version_output.structured.version.chassis_sn }}"
      - "Uptime: {{ version_output.structured.version.uptime }}"

- name: 4. show processes cpu 커맨드 결과값 가져오기 (Get the output of the co
  pyats_parse_command:
    command: show processes cpu
    register: cpu_output

- name: 5. CPU 5분 평균 사용률 보여주기 (Print out the average cpu usage(5 min
  debug:
    msg:
      - "CPU: {{ cpu_output.structured.five_min_cpu }}%"
```

```
TASK [2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)] *****
ok: [cat9k-1] => {
  "msg": [
    {
      "version": {
        "air_license_level": "AIR DNA Advantage",
        "chassis": "C9KV-UADP-8P",
        "chassis_sn": "9M2ST6PVK0A",
        "compiled_by": "mcpres",
        "compiled_date": "Wed 21-Sep-22 22:33",
        "copyright_years": "1986-2022",
        "curr_config_register": "0x2102",
        "disks": {
          "bootflash:": {
            "disk_size": "5236224",
            "type_of_disk": "virtual hard disk"
          }
        },
        "hostname": "cat9kv1",
        "image_id": "CAT9K_IOSXE",
        "image_type": "production image",
        "label": "RELEASE SOFTWARE (fc1)",
        "last_reload_reason": "unknown reload cause - reason ptr 0xF, PC 0x0, address 0x0",
        "license_package": {
          "None": {
            "license_level": "None",
            "license_type": "Subscription Smart License",
            "next_reload_license_level": "None"
          },
          "network-essentials": {
            "license_level": "network-essentials",
            "license_type": "Smart License",
            "next_reload_license_level": "network-essentials"
          }
        },
        "location": "Dublin",
```

PyATS 예제 1) 스위치 정보 가져오기

```
- "Hostname: {{ version_output.structured.version.hostname }}"
```

TASK [2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)] *****

```
ok: [cat9k-1] => {
  "msg": [
    {
      "version": {
        "air_license_level": "AIR DNA Advantage",
        "chassis": "C9KV-UADP-8P",
        "chassis_sn": "9M2ST6PVK0A",
        "compiled_by": "mcpre",
        "compiled_date": "Wed 21-Sep-22 22:33",
        "copyright_years": "1986-2022",
        "curr_config_register": "0x2102",
        "disks": {
          "bootflash.": {
            "disk_size": "5236224",
            "type_of_disk": "virtual hard disk"
          }
        },
        "hostname": "cat9kv1",
        "image_id": "CAT9K_IOSXE",
```

PyATS 예제 1) 스위치 정보 가져오기

```
- "Hostname: {{ version_output.structured.version.hostname }}"
```

```
TASK [2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)] *****
ok: [cat9k-1] => {
  "msg": [
    {
      "version": {
        "str_license_level": "AIR DNA Advantage",
        "chassis": "C9KV-UADP-8P",
        "chassis_sn": "9M2ST6PVK0A",
        "compiled_by": "mcpre",
        "compiled_date": "Wed 21-Sep-22 22:33",
        "copyright_years": "1986-2022",
        "curr_config_register": "0x2102",
        "disks": {
          "bootflash:": {
            "disk_size": "5236224",
            "type_of_disk": "virtual hard disk"
          }
        }
      },
      "hostname": "cat9kv1",
      "image_id": "CAT9K_IOSXE",
    }
  ]
}
```

PyATS 예제 1) 스위치 정보 가져오기

```
- "Hostname: {{ version_output.structured.version.hostname }}"
```

```
TASK [2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)] *****
ok: [cat9k-1] => {
  "msg": [
    {
      "version": {
        "air_license_level": "AIR DNA Advantage",
        "chassis": "C9KV-UADP-8P",
        "chassis_sn": "9M2ST6PVK0A",
        "compiled_by": "mcpre",
        "compiled_date": "Wed 21-Sep-22 22:33",
        "copyright_years": "1986-2022",
        "curr_config_register": "0x2102",
        "disks": {
          "bootflash:": {
            "disk_size": "5236224",
            "type_of_disk": "virtual hard disk"
          }
        }
      },
      "hostname": "cat9kv1",
      "image_id": "CAT9K_IOSXE",
    }
  ]
}
```


PyATS 예제 1) 스위치 정보 가져오기

```
- "Hostname: {{ version_output.structured.version.hostname }}"
```

```
TASK [2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)] *****
ok: [cat9k-1] => {
  "msg": [
    {
      "version": {
        "air_license_level": "AIR DNA Advantage",
        "chassis": "C9KV-UADP-8P",
        "chassis_sn": "9M2ST6PVK0A",
        "compiled_by": "mcpre",
        "compiled_date": "Wed 21-Sep-22 22:33",
        "copyright_years": "1986-2022",
        "curr_config_register": "0x2102",
        "disks": {
          "bootflash:": {
            "disk_size": "5236224",
            "type_of_disk": "virtual hard disk"
          }
        }
      },
      "hostname": "cat9kv1",
      "image_id": "CAT9K_IOSXE",
    }
  ]
}
```


PyATS 예제 1) 스위치 정보 가져오기

```
- "Hostname: {{ version_output.structured.version.hostname }}"
```

TASK [2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)] *****

```
ok: [cat9k-1] => {
  "msg": [
    {
      "version": {
        "air_license_level": "AIR DNA Advantage",
        "chassis": "C9KV-UADP-8P",
        "chassis_sn": "9M2ST6PVKOA",
        "compiled_by": "mcpres",
        "compiled_date": "Wed 21-Sep-22 22:33",
        "copyright_years": "1986-2022",
        "curr_config_register": "0x2102",
        "disks": {
          "bootflash.": {
            "disk_size": "5236224",
            "type_of_disk": "virtual hard disk"
          }
        }
      },
      "hostname": "cat9kv1",
      "image_id": "CAT9K_IOSXE",
    }
  ]
}
```

TASK [3. 스위치 정보 보여주기 - Hostname, Version, Serial Number, Uptime] ***

```
ok: [cat9k-1] => {
  "msg": [
    {
      "Hostname": "cat9kv1",
      "Version": "17.10.1pr07",
      "Serial Number": "9M2ST6PVKOA",
      "Uptime": "4 days, 19 hours, 38 minutes"
    }
  ]
}
```

PyATS 예제 1) 스위치 정보 가져오기

```
tasks:
- name: 1. show version 커맨드 결과값 가져오기 (Get the output of the command "show version")
  pyats_parse_command:
    command: show version
    register: version_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)
  debug:
    msg:
      - "{{version_output.structured}}"

- name: 3. 스위치 정보 보여주기 - Hostname, Version, Serial Number, Uptime (Display Switch
  debug:
    msg:
      - "Hostname: {{ version_output.structured.version.hostname }}"
      - "Version: {{ version_output.structured.version.version }}"
      - "Serial Number: {{ version_output.structured.version.chassis_sn }}"
      - "Uptime: {{ version_output.structured.version.uptime }}"

- name: 4. show processes cpu 커맨드 결과값 가져오기 (Get the output of the command "show processes cpu")
  pyats_parse_command:
    command: show processes cpu
    register: cpu_output

- name: 5. CPU 5분 평균 사용률 보여주기 (Print out the average cpu usage(5 minutes))
  debug:
    msg:
      - "CPU: {{ cpu_output.structured.five_min_cpu }}"
```

```
TASK [3. 스위치 정보 보여주기 - Hostname, Version, Serial Number, Uptime] ***
ok: [cat9k-1] => {
  "msg": [
    "Hostname: cat9kv1",
    "Version: 17.10.1prd7",
    "Serial Number: 9M2ST6PVK0A",
    "Uptime: 4 days, 19 hours, 38 minutes"
  ]
}
```

PyATS 예제 1) 스위치 정보 가져오기

```
tasks:
- name: 1. show version 커맨드 결과값 가져오기 (Get the output of the command "show version")
  pyats_parse_command:
    command: show version
    register: version_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)
  debug:
    msg:
      - "{{version_output.structured}}"

- name: 3. 스위치 정보 보여주기 - Hostname, Version, Serial Number, Uptime (Display Switch Information - Hostname, Version, Serial Number, Uptime)
  debug:
    msg:
      - "Hostname: {{ version_output.structured.version.hostname }}"
      - "Version: {{ version_output.structured.version.version }}"
      - "Serial Number: {{ version_output.structured.version.chassis_sn }}"
      - "Uptime: {{ version_output.structured.version.uptime }}"

- name: 4. show processes cpu 커맨드 결과값 가져오기 (Get the output of the command "show processes cpu")
  pyats_parse_command:
    command: show processes cpu
    register: cpu_output

- name: 5. CPU 5분 평균 사용률 보여주기 (Print out the average cpu usage(5 minutes))
  debug:
    msg:
      - "CPU: {{ cpu_output.structured.five_min_cpu }}%"
```

PyATS 예제 1) 스위치 정보 가져오기

```
tasks:
- name: 1. show version 커맨드 결과값 가져오기 (Get the output of the command)
  pyats_parse_command:
    command: show version
    register: version_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)
  debug:
    msg:
      - "{{version_output.structured}}"

- name: 3. 스위치 정보 보여주기 - Hostname, Version, Serial Number, Uptime (Display switch information)
  debug:
    msg:
      - "Hostname: {{ version_output.structured.version.hostname }}"
      - "Version: {{ version_output.structured.version.version }}"
      - "Serial Number: {{ version_output.structured.version.chassis_sn }}"
      - "Uptime: {{ version_output.structured.version.uptime }}"

- name: 4. show processes cpu 커맨드 결과값 가져오기 (Get the output of the command)
  pyats_parse_command:
    command: show processes cpu
    register: cpu_output

- name: 5. CPU 5분 평균 사용률 보여주기 (Print out the average cpu usage(5 min))
  debug:
    msg:
      - "CPU: {{ cpu_output.structured.five_min_cpu }}"
```

Genie

APIs
CLEAN
MODELS
PARSERS
TRIGGERS
VERIFICATIONS

Parsers List

show processes cpu

OS: XE IOSXE

matched

| | |
|------------------------|-------------------------------|
| XE show processes cpu | XE show processes (processid) |
| IOS show processes cpu | |
| XR show processes cpu | XR show processes (process) |
| NX show processes cpu | |

suggested

| | |
|--|---|
| XE show processes cpu history | XE show processes cpu platform |
| XE show processes cpu platform sorted | XE show processes cpu sorted |
| XE show processes cpu sorted (sort_time) | XE show processes cpu sorted (sort_time) exclude (exclude) |
| XE show processes cpu sorted (sort_time) include (key_word) | XE show processes cpu sorted exclude (exclude) |
| XE show processes cpu sorted include (key_word) | IOS show processes cpu history |
| IOS show processes cpu platform | IOS show processes cpu sorted |
| IOS show processes cpu sorted (sort_time) | IOS show processes cpu sorted (sort_time) exclude (exclude) |
| IOS show processes cpu sorted (sort_time) include (key_word) | IOS show processes cpu sorted exclude (exclude) |
| IOS show processes cpu sorted include (key_word) | XR show processes |
| NX show processes | NX show processes cpu include (include) |

PyATS 예제 1) 스위치 정보 가져오기

```
tasks:
- name: 1. show version 커맨드 결과값 가져오기 (Get the output of the command "show version")
  pyats_parse_command:
    command: show version
    register: version_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)
  debug:
    msg:
      - "{{version_output.structured}}"

- name: 3. 스위치 정보 보여주기 - Hostname, Version, Serial Number, Uptime (Display switch information)
  debug:
    msg:
      - "Hostname: {{ version_output.structured.version.hostname }}"
      - "Version: {{ version_output.structured.version.version }}"
      - "Serial Number: {{ version_output.structured.version.chassis_sn }}"
      - "Uptime: {{ version_output.structured.version.uptime }}"

- name: 4. show processes cpu 커맨드 결과값 가져오기 (Get the output of the command "show processes cpu")
  pyats_parse_command:
    command: show processes cpu
    register: cpu_output

- name: 5. CPU 5분 평균 사용률 보여주기 (Print out the average cpu usage(5 minutes))
  debug:
    msg:
      - "CPU: {{ cpu_output.structured.five_min_cpu }}"
```

Genie

APIs
CLEAN
MODELS
PARSERS
TRIGGERS
VERIFICATIONS

show processes cpu

< Back

Parser for show processes cpu
show processes cpu | include <WORD>

Schema

```
{
  Optional (str) five_sec_cpu_interrupts: <class 'int'>,
  Optional (str) five_sec_cpu_total: <class 'int'>,
  Optional (str) one_min_cpu: <class 'int'>,
  Optional (str) five_min_cpu: <class 'int'>,
  Optional (str) zero_cpu_processes: <class 'list'>,
  Optional (str) nonzero_cpu_processes: <class 'list'>,
  Optional (str) sort: {
    Any (str) *: {
      'runtime': <class 'int'>,
      'invoked': <class 'int'>,
      'usecs': <class 'int'>,
      'five_sec_cpu': <class 'float'>,
      'one_min_cpu': <class 'float'>,
      'five_min_cpu': <class 'float'>,
      'tty': <class 'int'>,
      'pid': <class 'int'>,
      'process': <class 'str'>,
    },
  },
}
```

PyATS 예제 1) 스위치 정보 가져오기

```
tasks:
- name: 1. show version 커맨드 결과값 가져오기 (Get the output of the command "show version")
  pyats_parse_command:
    command: show version
    register: version_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)
  debug:
    msg:
      - "{{version_output.structured}}"

- name: 3. 스위치 정보 보여주기 - Hostname, Version, Serial Number, Uptime (Display switch information)
  debug:
    msg:
      - "Hostname: {{ version_output.structured.version.hostname }}"
      - "Version: {{ version_output.structured.version.version }}"
      - "Serial Number: {{ version_output.structured.version.chassis_sn }}"
      - "Uptime: {{ version_output.structured.version.uptime }}"

- name: 4. show processes cpu 커맨드 결과값 가져오기 (Get the output of the command "show processes cpu")
  pyats_parse_command:
    command: show processes cpu
    register: cpu_output

- name: 5. CPU 5분 평균 사용률 보여주기 (Print out the average cpu usage(5 minutes))
  debug:
    msg:
      - "CPU: {{ cpu_output.structured.five_min_cpu }}"
```

Genie

APIS
CLEAN
MODELS
PARSERS
TRIGGERS
VERIFICATIONS

show processes cpu

< Back

View Source

Parser for show processes cpu
show processes cpu | include <WORD>

Schema

```
{
  Optional (str) five_sec_cpu_interrupts: <class 'int'>,
  Optional (str) five_sec_cpu_total: <class 'int'>,
  Optional (str) one_min_cpu: <class 'int'>,
  Optional (str) five_min_cpu: <class 'int'>,
  Optional (str) zero_cpu_processes: <class 'list'>,
  Optional (str) nonzero_cpu_processes: <class 'list'>,
  Optional (str) sort: {
    Any (str) *: {
      'runtime': <class 'int'>,
      'invoked': <class 'int'>,
      'usesec': <class 'int'>,
      'five_sec_cpu': <class 'float'>,
      'one_min_cpu': <class 'float'>,
      'five_min_cpu': <class 'float'>,
      'tty': <class 'int'>,
      'pid': <class 'int'>,
      'process': <class 'str'>,
    },
  },
}
```


PyATS 예제 1) 스위치 정보 가져오기

```
tasks:
- name: 1. show version 커맨드 결과값 가져오기 (Get the output of the command "show version")
  pyats_parse_command:
    command: show version
    register: version_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)
  debug:
    msg:
      - "{{version_output.structured}}"

- name: 3. 스위치 정보 보여주기 - Hostname, Version, Serial Number, Uptime (Display switch information)
  debug:
    msg:
      - "Hostname: {{ version_output.structured.version.hostname }}"
      - "Version: {{ version_output.structured.version.version }}"
      - "Serial Number: {{ version_output.structured.version.chassis_sn }}"
      - "Uptime: {{ version_output.structured.version.uptime }}"

- name: 4. show processes cpu 커맨드 결과값 가져오기 (Get the output of the command "show processes cpu")
  pyats_parse_command:
    command: show processes cpu
    register: cpu_output

- name: 5. CPU 5분 평균 사용률 보여주기 (Print out the average cpu usage(5 minutes))
  debug:
    msg:
      - "CPU: {{ cpu_output.structured.five_min_cpu }}%"
```

Genie

show processes cpu

< Back

Parser for show processes cpu
show processes cpu | include <WORD>

Schema

```
{
  Optional (str) five_sec_cpu_interrupts: <class 'int'>,
  Optional (str) five_sec_cpu_total: <class 'int'>,
  Optional (str) one_min_cpu: <class 'int'>,
  Optional (str) five_min_cpu: <class 'int'>,
  Optional (str) zero_cpu_processes: <class 'list'>,
  Optional (str) nonzero_cpu_processes: <class 'list'>,
  Optional (str) sort: {
    Any (str) *: {
      'runtime': <class 'int'>,
      'invoked': <class 'int'>,
      'usesec': <class 'int'>,
      'five_sec_cpu': <class 'float'>,
      'one_min_cpu': <class 'float'>,
      'five_min_cpu': <class 'float'>,
      'tty': <class 'int'>,
      'pid': <class 'int'>,
      'process': <class 'str'>,
    },
  },
}
```

PyATS 예제 1) 스위치 정보 가져오기

```
tasks:
- name: 1. show version 커맨드 결과값 가져오기 (Get the output of the command "show version")
  pyats_parse_command:
    command: show version
    register: version_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable version_output)
  debug:
    msg:
      - "{{version_output.structured}}"

- name: 3. 스위치 정보 보여주기 - Hostname, Version, Serial Number, Uptime (Display Switch Information - Hostname, Version, Serial Number, Uptime)
  debug:
    msg:
      - "Hostname: {{ version_output.structured.version.hostname }}"
      - "Version: {{ version_output.structured.version.version }}"
      - "Serial Number: {{ version_output.structured.version.chassis_sn }}"
      - "Uptime: {{ version_output.structured.version.uptime }}"

- name: 4. show processes cpu 커맨드 결과값 가져오기 (Get the output of the command "show processes cpu")
  pyats_parse_command:
    command: show processes cpu
    register: cpu_output

- name: 5. CPU 5분 평균 사용률 보여주기 (Print out the average cpu usage(5 minutes))
  debug:
    msg:
      - "CPU: {{ cpu_output.structured.five_min_cpu }}%"
```

```
TASK [5. CPU 5분 평균 사용률 보여주기 (Print out the average cpu usage(5 minutes))]
ok: [cat9k-1] => {
  "msg": [
    "CPU: 1%"
  ]
}
```


PyATS 예제 2) interface 개수 확인하기

[입력 필요]

1) ansible-playbook
02_count_up_interface.yaml

목표

- 1) GigabitEthernet 인터페이스 개수 세기
- 2) GigabitEthernet & up 상태인 인터페이스 개수 세기

방법...?

- 1) 모든 인터페이스를 가져오는 방법...?
- 2) 필터링을 하는 방법..?

실행방법

```
devnet@devnet-virtual-machine:~/DevNet_Korea/02_IOSXE$ ansible-playbook 02_count_up_interface.yaml
```

PyATS 예제 2) interface 개수 확인하기

```
tasks:
- name: 1. show interfaces 커맨드 결과값 가져오기 (Get the output of the command "show interfaces")
  pyats_parse_command:
    command: show interfaces
    register: interface_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable interface_output)
  debug:
    msg: "{{ interface_output.structured }}"

- name: 3. 인터페이스 type, status 만 저장하기 (Save the type & status of the interface)
  set_fact:
    interface_type_status: "{{ interface_output | json_query('structured.*.{type : type, oper_status : oper_status}') }}"

- name: 4. 인터페이스 type, status 보여주기 (Show the variable interface_type_status)
  debug:
    msg: "{{ interface_type_status }}"

- name: 5. GigabitEthernet 인터페이스만 저장하기 (Save the interface which type is gigabit ethernet)
  set_fact:
    interface_gigethernet: "{{ interface_type_status | json_query('[?type == `Gigabit Ethernet`']') }}"

- name: 6. GigabitEthernet 인터페이스 보여주기
  debug:
    msg: "{{ interface_gigethernet }}"

- name: 7. GigabitEthernet이면서 up 상태인 인터페이스만 저장하기 (Save the interface which is GigabitEthernet and up)
  set_fact:
    interface_up: "{{ interface_gigethernet | json_query('[?oper_status == `up`']') }}"

- name: 8. GigabitEthernet이면서 up 상태인 인터페이스만 보여주기 (Show the variable interface_up)
  debug:
    msg: "{{ interface_up }}"

- name: 9. 총 인터페이스 수 & up 인터페이스 수 보여주기 (Print out the total number of Interfaces & Up interfaces)
  debug:
    msg:
      - "Total number of Gigabit Ethernet Interfaces : {{ interface_gigethernet | length }}"
      - "Total number of Gigabit Ethernet & Up Interfaces : {{ interface_up | length }}"
```

PyATS 예제 2) interface 개수 확인하기

```
tasks:
- name: 1. show interfaces 커맨드 결과값 가져오기 (Get the output of the command "show interfaces")
  pyats_parse_command:
    command: show interfaces
    register: interface_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable interface_output)
  debug:
    msg: "{{ interface_output.structured }}"

- name: 3. 인터페이스 type, status 만 저장하기 (Save the type & status of the interface)
  set_fact:
    interface_type_status: "{{ interface_output | json_query('structured.*.{type : type, oper_status : oper_status}') }}"

- name: 4. 인터페이스 type, status 보여주기 (Show the variable interface_type_status)
  debug:
    msg: "{{ interface_type_status }}"

- name: 5. GigabitEthernet 인터페이스만 저장하기 (Save the interface which type is gigabit ethernet)
  set_fact:
    interface_gigethernet: "{{ interface_type_status | json_query('[?type == `Gigabit Ethernet`'] ) }}"

- name: 6. GigabitEthernet 인터페이스 보여주기
  debug:
    msg: "{{ interface_gigethernet }}"

- name: 7. GigabitEthernet이면서 up 상태인 인터페이스만 저장하기 (Save the interface which is GigabitEthernet and up)
  set_fact:
    interface_up: "{{ interface_gigethernet | json_query('[?oper_status == `up`'] ) }}"

- name: 8. GigabitEthernet이면서 up 상태인 인터페이스만 보여주기 (Show the variable interface_up)
  debug:
    msg: "{{ interface_up }}"

- name: 9. 총 인터페이스 수 & up 인터페이스 수 보여주기 (Print out the total number of Interfaces & Up interfaces)
  debug:
    msg:
      - "Total number of Gigabit Ethernet Interfaces : {{ interface_gigethernet | length }}"
      - "Total number of Gigabit Ethernet & Up Interfaces : {{ interface_up | length }}"
```

```
    "type": "RP management port"
  },
  "GigabitEthernet1/0/1": {
    "arp_timeout": "04:00:00",
    "arp_type": "arpa",
    "auto_negotiate": true,
    "bandwidth": 1000000,
    "connected": true,
    "counters": {
      "in_broadcast_pkts": 32242,
      "in_crc_errors": 0,
      "in_errors": 0,
      "in_frame": 0,
      "in_giants": 0,
      "in_ignored": 0,
      "in_mac_pause_frames": 0,
      "in_multicast_pkts": 29993,
      "in_no_buffer": 0,
      "in_octets": 7568686,
      "in_overrun": 0,
      "in_pkts": 32254,
      "in_runt": 0,
      "in_throttles": 0,
      "in_watchdog": 0,
      "in_with_dribble": 0,
      "last_clear": "never",
      "out_babble": 0,
      "out_broadcast_pkts": 176,
      "out_buffer_failure": 0,
      "out_buffers_swapped": 0,
      "out_collision": 0,
      "out_deferred": 0,
      "out_errors": 0,
      "out_interface_resets": 2,
      "out_late_collision": 0,
      "out_lost_carrier": 0,
      "out_mac_pause_frames": 0,
      "out_multicast_pkts": 51735,
      "out_no_carrier": 0,
      "out_octets": 4084837,
      "out_pkts": 51934,
      "out_underruns": 0,
      "out_unknown_protocol_drops": 0,
      "rate": {
        "in_rate": 0,
        "in_rate_pkts": 0,
        "load_interval": 300,
        "out_rate": 0,
        "out_rate_pkts": 0
      }
    },
    "delay": 10,
    "duplex_mode": "full",
    "enabled": true,
    "encapsulations": {
      "encapsulation": "arpa"
    },
    "err_disabled": false,
    "flow_control": {
      "receive": true,
      "send": false
    },
    "is_deleted": false,
    "last_input": "00:00:05",
```

PyATS 예제 2) interface 개수 확인하기

```
tasks:
- name: 1. show interfaces 커맨드 결과값 가져오기 (Get the output of the command "show interfaces")
  pyats_parse_command:
    command: show interfaces
    register: interface_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable interface_output)
  debug:
    msg: "{{ interface_output.structured }}"

- name: 3. 인터페이스 type, status 만 저장하기 (Save the type & status of the interface)
  set_fact:
    interface_type_status: "{{ interface_output | json_query('structured.*.{type : type, oper_status : oper_status}') }}"

- name: 4. 인터페이스 type, status 보여주기 (Show the variable interface_type_status)
  debug:
    msg: "{{ interface_type_status }}"

- name: 5. GigabitEthernet 인터페이스만 저장하기 (Save the interface which type is gigabit ethernet)
  set_fact:
    interface_gigethernet: "{{ interface_type_status | json_query('[?type == `Gigabit Ethernet`']') }}"

- name: 6. GigabitEthernet 인터페이스 보여주기
  debug:
    msg: "{{ interface_gigethernet }}"

- name: 7. GigabitEthernet이면서 up 상태인 인터페이스만 저장하기 (Save the interface which is GigabitEthernet and up)
  set_fact:
    interface_up: "{{ interface_gigethernet | json_query('[?oper_status == `up`']') }}"

- name: 8. GigabitEthernet이면서 up 상태인 인터페이스만 보여주기 (Show the variable interface_up)
  debug:
    msg: "{{ interface_up }}"

- name: 9. 총 인터페이스 수 & up 인터페이스 수 보여주기 (Print out the total number of Interfaces & Up interfaces)
  debug:
    msg:
      - "Total number of Gigabit Ethernet Interfaces : {{ interface_gigethernet | length }}"
      - "Total number of Gigabit Ethernet & Up Interfaces : {{ interface_up | length }}"
```

3번 task의 목표 :

모든 인터페이스의 정보 중에서
type 과 status 에 대한 정보만
뽑아서 저장하기!

PyATS 예제 2) interface 개수 확인하기

```
tasks:
- name: 1. show interfaces 커맨드 결과값 가져오기 (Get the output of the command "show interfaces")
  pyats_parse_command:
    command: show interfaces
    register: interface_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable interface_output)
  debug:
    msg: "{{ interface_output.structured }}"

- name: 3. 인터페이스 type, status 만 저장하기 (Save the type & status of the interface)
  set_fact:
    interface_type_status: "{{ interface_output | json_query('structured.*.{type : type, oper_status : oper_status}') }}"

- name: 4. 인터페이스 type, status 보여주기 (Show the variable interface_type_status)
  debug:
    msg: "{{ interface_type_status }}"

- name: 5. GigabitEthernet 인터페이스만 저장하기 (Save the interface which type is gigabit ethernet)
  set_fact:
    interface_gigethernet: "{{ interface_type_status | json_query('[?type == `Gigabit Ethernet`']') }}"

- name: 6. GigabitEthernet 인터페이스 보여주기
  debug:
    msg: "{{ interface_gigethernet }}"

- name: 7. GigabitEthernet이면서 up 상태인 인터페이스만 저장하기 (Save the interface which is GigabitEthernet and up)
  set_fact:
    interface_up: "{{ interface_gigethernet | json_query('[?oper_status == `up`']') }}"

- name: 8. GigabitEthernet이면서 up 상태인 인터페이스만 보여주기 (Show the variable interface_up)
  debug:
    msg: "{{ interface_up }}"

- name: 9. 총 인터페이스 수 & up 인터페이스 수 보여주기 (Print out the total number of Interfaces & Up interfaces)
  debug:
    msg:
      - "Total number of Gigabit Ethernet Interfaces : {{ interface_gigethernet | length }}"
      - "Total number of Gigabit Ethernet & Up Interfaces : {{ interface_up | length }}"
```

필터링

- `{{ 데이터 | 필터 }}`
- 데이터에 필터를 장착해서 원하는 모양으로 변경!
- 여러가지 필터가 있음
- https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_filters.html

PyATS 예제 2) interface 개수 확인하기

```
tasks:
- name: 1. show interfaces 커맨드 결과값 가져오기 (Get the output of the command "show interfaces")
  pyats_parse_command:
    command: show interfaces
    register: interface_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable interface_output)
  debug:
    msg: "{{ interface_output.structured }}"

- name: 3. 인터페이스 type, status 만 저장하기 (Save the type & status of the interface)
  set_fact:
    interface_type_status: "{{ interface_output | json_query('structured.*.{type : type, oper_status : oper_status}') }}"

- name: 4. 인터페이스 type, status 보여주기 (Show the variable interface_type_status)
  debug:
    msg: "{{ interface_type_status }}"

- name: 5. GigabitEthernet 인터페이스만 저장하기 (Save the interface which type is gigabit ethernet)
  set_fact:
    interface_gigethernet: "{{ interface_type_status | json_query('[?type == `Gigabit Ethernet`'] ) }}"

- name: 6. GigabitEthernet 인터페이스 보여주기
  debug:
    msg: "{{ interface_gigethernet }}"

- name: 7. GigabitEthernet이면서 up 상태인 인터페이스만 저장하기 (Save the interface which is GigabitEthernet and up)
  set_fact:
    interface_up: "{{ interface_gigethernet | json_query('[?oper_status == `up`'] ) }}"

- name: 8. GigabitEthernet이면서 up 상태인 인터페이스만 보여주기 (Show the variable interface_up)
  debug:
    msg: "{{ interface_up }}"

- name: 9. 총 인터페이스 수 & up 인터페이스 수 보여주기 (Print out the total number of Interfaces & Up interfaces)
  debug:
    msg:
      - "Total number of Gigabit Ethernet Interfaces : {{ interface_gigethernet | length }}"
      - "Total number of Gigabit Ethernet & Up Interfaces : {{ interface_up | length }}"
```

우리가 쓸 필터: Json_query

= json 형태의 값을 query할 수 있게 해주는 필터

= { {}, {}, {{}}, {} }.. 이런 값을 돌면서 필요한 값을 찾을 수 있게 해주는 필터

참고:

https://docs.ansible.com/ansible/latest/collections/community/general/json_query_filter.html

PyATS 예제 2) interface 개수 확인하기

```
tasks:
- name: 1. show interfaces 커맨드 결과값 가져오기 (Get the output of the command "show interfaces")
  pyats_parse_command:
    command: show interfaces
    register: interface_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable interface_output)
  debug:
    msg: "{{ interface_output.structured }}"

- name: 3. 인터페이스 type, status 만 저장하기 (Save the type & status of the interface)
  set_fact:
    interface_type_status: "{{ interface_output | json_query('structured.*.{type : type, oper_status : oper_status}') }}"

- name: 4. 인터페이스 type, status 보여주기 (Show the variable interface_type_status)
  debug:
    msg: "{{ interface_type_status }}"

- name: 5. GigabitEthernet 인터페이스만 저장하기 (Save the interface which type is gigabit ethernet)
  set_fact:
    interface_gigethernet: "{{ interface_type_status | json_query('[?type == `Gigabit Ethernet`']') }}"

- name: 6. GigabitEthernet 인터페이스 보여주기
  debug:
    msg: "{{ interface_gigethernet }}"

- name: 7. GigabitEthernet이면서 up 상태인 인터페이스만 저장하기 (Save the interface which is GigabitEthernet and up)
  set_fact:
    interface_up: "{{ interface_gigethernet | json_query('[?oper_status == `up`']') }}"

- name: 8. GigabitEthernet이면서 up 상태인 인터페이스만 보여주기 (Show the variable interface_up)
  debug:
    msg: "{{ interface_up }}"

- name: 9. 총 인터페이스 수 & up 인터페이스 수 보여주기 (Print out the total number of Interfaces & Up interfaces)
  debug:
    msg:
      - "Total number of Gigabit Ethernet Interfaces : {{ interface_gigethernet | length }}"
      - "Total number of Gigabit Ethernet & Up Interfaces : {{ interface_up | length }}"
```

3번 task의 목표 :

모든 인터페이스의 정보 중에서
type 과 status 에 대한 정보만
뽑아서 저장하기!

방법:

- 1) *를 통해서 모든 인터페이스의 정보를 가져온다
- 2) type 정보는 type이라는 key, oper_status 정보는 oper_status라는 키로 저장

PyATS 예제 2) interface 개수 확인하기

```
tasks:
- name: 1. show interfaces 커맨드 결과값 가져오기 (Get the output of the command 'show interfaces')
  pyats_parse_command:
    command: show interfaces
    register: interface_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable interface_output)
  debug:
    msg: "{{ interface_output.structured }}"

- name: 3. 인터페이스 type, status 만 저장하기 (Save the type & status of the interface)
  set_fact:
    interface_type_status: "{{ interface_output | json_query('structured.*.type') }}"

- name: 4. 인터페이스 type, status 보여주기 (Show the variable interface_type_status)
  debug:
    msg: "{{ interface_type_status }}"

- name: 5. GigabitEthernet 인터페이스만 저장하기 (Save the interface which type is GigabitEthernet)
  set_fact:
    interface_gigetherenet: "{{ interface_type_status | json_query('[?type == \"GigabitEthernet\"]') }}"

- name: 6. GigabitEthernet 인터페이스 보여주기
  debug:
    msg: "{{ interface_gigetherenet }}"

- name: 7. GigabitEthernet이면서 up 상태인 인터페이스만 저장하기 (Save the interface which type is GigabitEthernet and status is up)
  set_fact:
    interface_up: "{{ interface_gigetherenet | json_query('[?oper_status == \"up\"]') }}"

- name: 8. GigabitEthernet이면서 up 상태인 인터페이스만 보여주기 (Show the variable interface_up)
  debug:
    msg: "{{ interface_up }}"

- name: 9. 총 인터페이스 수 & up 인터페이스 수 보여주기 (Print out the total number of interfaces)
  debug:
    msg:
      - "Total number of Gigabit Ethernet Interfaces : {{ interface_gigetherenet | length }}"
      - "Total number of Gigabit Ethernet & Up Interfaces : {{ interface_up | length }}
```

```
    },
    "type": "RP management port"
  },
  "GigabitEthernet1/0/1": {
    "arp_timeout": "04:00:00",
    "arp_type": "arpa",
    "auto_negotiate": true,
    "bandwidth": 1000000,
    "connected": true,
    "counters": {
      "in_broadcast_pkts": 32242,
      "in_crc_errors": 0,
      "in_errors": 0,
      "in_frame": 0,
      "in_giants": 0,
      "in_ignored": 0,
      "in_mac_pause_frames": 0,
      "in_multicast_pkts": 29993,
      "in_no_buffer": 0,
      "in_octets": 7568686,
      "in_overrun": 0,
      "in_pkts": 32254,
      "in_runs": 0,
      "in_throttles": 0,
      "in_watchdog": 0,
      "in_with_dribble": 0,
      "last_clear": "never",
      "out_babble": 0,
      "out_broadcast_pkts": 176,
      "out_buffer_failure": 0,
      "out_buffers_swapped": 0,
      "out_collision": 0,
      "out_deferred": 0,
      "out_errors": 0,
      "out_interface_resets": 2,
      "out_late_collision": 0,
      "out_lost_carrier": 0,
      "out_mac_pause_frames": 0,
      "out_multicast_pkts": 51735,
      "out_no_carrier": 0,
      "out_octets": 4084837,
      "out_pkts": 51934,
      "out_underruns": 0,
      "out_unknown_protocol_drops": 0,
      "rate": {
        "in_rate": 0,
        "in_rate_pkts": 0,
        "load_interval": 300,
        "out_rate": 0,
        "out_rate_pkts": 0
      }
    },
    "delay": 10,
    "duplex_mode": "full",
    "enabled": true,
    "encapsulations": {
      "encapsulation": "arpa"
    },
    "err_disabled": false,
    "flow_control": {
      "receive": true,
      "send": false
    },
    "is_deleted": false,
    "last_input": "00:00:00",
```



```
ok: [cat9k-1] => {
  "msg": [
    {
      "oper_status": "down",
      "type": "Ethernet SVI"
    },
    {
      "oper_status": "up",
      "type": "RP management port"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    }
  ]
}
```


PyATS 예제 2) interface 개수 확인하기

```
tasks:
- name: 1. show interfaces 커맨드 결과값 가져오기 (Get the output of the command "show interfaces")
  pyats_parse_command:
    command: show interfaces
    register: interface_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable interface_output)
  debug:
    msg: "{{ interface_output.structured }}"

- name: 3. 인터페이스 type, status 만 저장하기 (Save the type & status of the interface)
  set_fact:
    interface_type_status: "{{ interface_output | json_query('structured.*.{type : type, oper_status : oper_status}') }}"

- name: 4. 인터페이스 type, status 보여주기 (Show the variable interface_type_status)
  debug:
    msg: "{{ interface_type_status }}"

- name: 5. GigabitEthernet 인터페이스만 저장하기 (Save the interface which type is gigabit ethernet)
  set_fact:
    interface_gigethernet: "{{ interface_type_status | json_query('[?type == `Gigabit Ethernet`]') }}"

- name: 6. GigabitEthernet 인터페이스 보여주기
  debug:
    msg: "{{ interface_gigethernet }}"

- name: 7. GigabitEthernet이면서 up 상태인 인터페이스만 저장하기 (Save the interface which is GigabitEthernet and up)
  set_fact:
    interface_up: "{{ interface_gigethernet | json_query('[?oper_status == `up`]') }}"

- name: 8. GigabitEthernet이면서 up 상태인 인터페이스만 보여주기 (Show the variable interface_up)
  debug:
    msg: "{{ interface_up }}"

- name: 9. 총 인터페이스 수 & up 인터페이스 수 보여주기 (Print out the total number of Interfaces & Up interfaces)
  debug:
    msg:
      - "Total number of Gigabit Ethernet Interfaces : {{ interface_gigethernet | length }}"
      - "Total number of Gigabit Ethernet & Up Interfaces : {{ interface_up | length }}"
```

5번 태스크의 목표 :
Type 이 Gigabit Ethernet 인
인터페이스만 뽑아오기!

방법:
조건을 달아보기

```
json_query('[?type == `Gigabit Ethernet`]') }}
```

조건

조건 문법 참고:

<https://jmespath.org/specification.html>

PyATS 예제 2) interface 개수 확인하기

```
tasks:
- name: 1. show interfaces 커맨드 결과값 가져오기 (Get the output of the command)
  pyats_parse_command:
    command: show interfaces
    register: interface_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable interface_output)
  debug:
    msg: "{{ interface_output.structured }}"

- name: 3. 인터페이스 type, status 만 저장하기 (Save the type & status of the interface)
  set_fact:
    interface_type_status: "{{ interface_output | json_query('structured') }}"

- name: 4. 인터페이스 type, status 보여주기 (Show the variable interface_type_status)
  debug:
    msg: "{{ interface_type_status }}"

- name: 5. GigabitEthernet 인터페이스만 저장하기 (Save the interface which type is GigabitEthernet)
  set_fact:
    interface_gigethernet: "{{ interface_type_status | json_query('[?type==GigabitEthernet]') }}"

- name: 6. GigabitEthernet 인터페이스 보여주기 (Show the variable interface_gigethernet)
  debug:
    msg: "{{ interface_gigethernet }}"

- name: 7. GigabitEthernet이면서 up 상태인 인터페이스만 저장하기 (Save the interface which type is GigabitEthernet and status is up)
  set_fact:
    interface_up: "{{ interface_gigethernet | json_query('[?oper_status==up]') }}"

- name: 8. GigabitEthernet이면서 up 상태인 인터페이스만 보여주기 (Show the variable interface_up)
  debug:
    msg: "{{ interface_up }}"

- name: 9. 총 인터페이스 수 & up 인터페이스 수 보여주기 (Print out the total number of interfaces and up interfaces)
  debug:
    msg:
      - "Total number of Gigabit Ethernet Interfaces : {{ interface_gigethernet | count }}"
      - "Total number of Gigabit Ethernet & Up Interfaces : {{ interface_up | count }}
```

```
ok: [cat9k-1] => {
  "msg": [
    {
      "oper_status": "down",
      "type": "Ethernet SVI"
    },
    {
      "oper_status": "up",
      "type": "RP management port"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    }
  ]
}
```



```
TASK [6. GigabitEthernet 인터페이스 보여주기] ***
*****
ok: [cat9k-1] => {
  "msg": [
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    }
  ]
}
```

PyATS 예제 2) interface 개수 확인하기

```
tasks:
- name: 1. show interfaces 커맨드 결과값 가져오기 (Get the output of the command "show interfaces")
  pyats_parse_command:
    command: show interfaces
    register: interface_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable interface_output)
  debug:
    msg: "{{ interface_output.structured }}"

- name: 3. 인터페이스 type, status 만 저장하기 (Save the type & status of the interface)
  set_fact:
    interface_type_status: "{{ interface_output | json_query('structured.*.{type : type, oper_status : oper_status}') }}"

- name: 4. 인터페이스 type, status 보여주기 (Show the variable interface_type_status)
  debug:
    msg: "{{ interface_type_status }}"

- name: 5. GigabitEthernet 인터페이스만 저장하기 (Save the interface which type is gigabit ethernet)
  set_fact:
    interface_gigethernet: "{{ interface_type_status | json_query('[?type == `Gigabit Ethernet`'] ) }}"

- name: 6. GigabitEthernet 인터페이스 보여주기
  debug:
    msg: "{{ interface_gigethernet }}"

- name: 7. GigabitEthernet이면서 up 상태인 인터페이스만 저장하기 (Save the interface which is GigabitEthernet and up)
  set_fact:
    interface_up: "{{ interface_gigethernet | json_query('[?oper_status == `up`'] ) }}"

- name: 8. GigabitEthernet이면서 up 상태인 인터페이스만 보여주기 (Show the variable interface_up)
  debug:
    msg: "{{ interface_up }}"

- name: 9. 총 인터페이스 수 & up 인터페이스 수 보여주기 (Print out the total number of Interfaces & Up interfaces)
  debug:
    msg:
      - "Total number of Gigabit Ethernet Interfaces : {{ interface_gigethernet | length }}"
      - "Total number of Gigabit Ethernet & Up Interfaces : {{ interface_up | length }}"
```

7번 태스크의 목표 :
Type 이 Gigabit Ethernet 인
인터페이스중 up인 인터페이스
뽑아보기!

방법:
조건을 달아보기

```
json_query('[?oper_status == `up`']')
```

조건

PyATS 예제 2) interface 개수 확인하기

```
tasks:
- name: 1. show interfaces 커맨드 결과값 가져오기 (Get the output of the command)
  pyats_parse_command:
    command: show interfaces
    register: interface_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable interface_output)
  debug:
    msg: "{{ interface_output.structured }}"

- name: 3. 인터페이스 type, status 만 저장하기 (Save the type & status of the interface)
  set_fact:
    interface_type_status: "{{ interface_output | json_query('structured') }}"

- name: 4. 인터페이스 type, status 보여주기 (Show the variable interface_type_status)
  debug:
    msg: "{{ interface_type_status }}"

- name: 5. GigabitEthernet 인터페이스만 저장하기 (Save the interface which type is GigabitEthernet)
  set_fact:
    interface_gigether_net: "{{ interface_type_status | json_query('[?type==GigabitEthernet]') }}"

- name: 6. GigabitEthernet 인터페이스 보여주기
  debug:
    msg: "{{ interface_gigether_net }}"

- name: 7. GigabitEthernet이면서 up 상태인 인터페이스만 저장하기 (Save the interface which type is GigabitEthernet and status is up)
  set_fact:
    interface_up: "{{ interface_gigether_net | json_query('[?oper_status==up]') }}"

- name: 8. GigabitEthernet이면서 up 상태인 인터페이스만 보여주기 (Show the variable interface_up)
  debug:
    msg: "{{ interface_up }}"

- name: 9. 총 인터페이스 수 & up 인터페이스 수 보여주기 (Print out the total number of interfaces and up interfaces)
  debug:
    msg:
      - "Total number of Gigabit Ethernet Interfaces : {{ interface_gigether_net | count }}"
      - "Total number of Gigabit Ethernet & Up Interfaces : {{ interface_up | count }}
```

```
ok: [cat9k-1] => {
  "msg": [
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    }
  ]
}
```



```
ok: [cat9k-1] => {
  "msg": [
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    }
  ]
}
```

PyATS 예제 2) interface 개수 확인하기

```
tasks:
- name: 1. show interfaces 커맨드 결과값 가져오기 (Get the output of the command "show interfaces")
  pyats_parse_command:
    command: show interfaces
    register: interface_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable interface_output)
  debug:
    msg: "{{ interface_output.structured }}"

- name: 3. 인터페이스 type, status 만 저장하기 (Save the type & status of the interface)
  set_fact:
    interface_type_status: "{{ interface_output | json_query('structured.*.{type : type, oper_status : oper_status}') }}"

- name: 4. 인터페이스 type, status 보여주기 (Show the variable interface_type_status)
  debug:
    msg: "{{ interface_type_status }}"

- name: 5. GigabitEthernet 인터페이스만 저장하기 (Save the interface which type is gigabit ethernet)
  set_fact:
    interface_gigethernet: "{{ interface_type_status | json_query('[?type == `Gigabit Ethernet`']') }}"

- name: 6. GigabitEthernet 인터페이스 보여주기
  debug:
    msg: "{{ interface_gigethernet }}"

- name: 7. GigabitEthernet이면서 up 상태인 인터페이스만 저장하기 (Save the interface which is GigabitEthernet and up)
  set_fact:
    interface_up: "{{ interface_gigethernet | json_query('[?oper_status == `up`']') }}"

- name: 8. GigabitEthernet이면서 up 상태인 인터페이스만 보여주기 (Show the variable interface_up)
  debug:
    msg: "{{ interface_up }}"

- name: 9. 총 인터페이스 수 & up 인터페이스 수 보여주기 (Print out the total number of Interfaces & Up interfaces)
  debug:
    msg:
      - "Total number of Gigabit Ethernet Interfaces : {{ interface_gigethernet | length }}"
      - "Total number of Gigabit Ethernet & Up Interfaces : {{ interface_up | length }}"
```

필터 length
- 변수 안에 개수를 세는 필터

```
ok: [cat9k-1] => {
  "msg": [
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    },
    {
      "oper_status": "up",
      "type": "Gigabit Ethernet"
    }
  ]
}
```

총 8개

PyATS 예제 2) interface 개수 확인하기

```
tasks:
- name: 1. show interfaces 커맨드 결과값 가져오기 (Get the output of the command "show interfaces")
  pyats_parse_command:
    command: show interfaces
    register: interface_output

- name: 2. 파싱된 커맨드 아웃풋 보여주기 (Print out the variable interface_output)
  debug:
    msg: "{{ interface_output.structured }}"

- name: 3. 인터페이스 type, status 만 저장하기 (Save the type & status of the interface)
  set_fact:
    interface_type_status: "{{ interface_output | json_query('structured.*.{type : type, oper_status : oper_status}') }}"

- name: 4. 인터페이스 type, status 보여주기 (Show the variable interface_type_status)
  debug:
    msg: "{{ interface_type_status }}"

- name: 5. GigabitEthernet 인터페이스만 저장하기 (Save the interface which type is gigabit ethernet)
  set_fact:
    interface_gigethernet: "{{ interface_type_status | json_query('[?type == `Gigabit Ethernet`']') }}"

- name: 6. GigabitEthernet 인터페이스 보여주기
  debug:
    msg: "{{ interface_gigethernet }}"

- name: 7. GigabitEthernet이면서 up 상태인 인터페이스만 저장하기 (Save the interface which is GigabitEthernet and up)
  set_fact:
    interface_up: "{{ interface_gigethernet | json_query('[?oper_status == `up`']') }}"

- name: 8. GigabitEthernet이면서 up 상태인 인터페이스만 보여주기 (Show the variable interface_up)
  debug:
    msg: "{{ interface_up }}"

- name: 9. 총 인터페이스 수 & up 인터페이스 수 보여주기 (Print out the total number of Interfaces & Up interfaces)
  debug:
    msg:
      - "Total number of Gigabit Ethernet Interfaces : {{ interface_gigethernet | length }}"
      - "Total number of Gigabit Ethernet & Up Interfaces : {{ interface_up | length }}"
```

```
TASK [9. 총 인터페이스 수 & up 인터페이스 수 보여주기 (Print out the total number of Interfaces & Up interfaces)]
ok: [cat9k-1] => {
  "msg": [
    "Total number of Gigabit Ethernet Interfaces : 8",
    "Total number of Gigabit Ethernet & Up Interfaces : 8"
  ]
}
```


미션!



미션 셋업



[미션 처음 한번만 세팅 필요]

- 1) sudo su
- 2) 1234Qwer

```
devnet@devnet-virtual-machine:~/DevNet_Korea/02_IOSXE$ sudo su  
[sudo] password for devnet:
```

미션 셋업



[계속 필요한 리눅스 커맨드 리스트]

- 미션 파일 수정하기

- 1) `vim 02_mission.yaml` (미션 파일로 들어가기)
- 2) `i` (미션 파일 수정하기 전에 입력하기)
- 3) `:wq!` (미션 파일 수정 후 저장하기)

- 미션 파일 실행하기

- 1) `ansible-playbook 02_mission.yaml`

미션 내용 (각 5점!)



1. Json_query를 사용하여 interface_output을
{type:type값, status:status값, mtu:mtu값}
형태로 변경해주세요!
(!!MISSION!! 부분을 지우고 수정해주세요!)

```
- name: 3. 인터페이스 type, status, mtu 저장하기 (Save the type & status & mtu of the interface)
  set_fact:
    interface_type_status: "{{ interface_output | json_query('structured.*.{!! MISSION !!}') }}"
```

##힌트

{type:type값, oper_status:oper_status값}으로
변경하는 방법

```
json_query('structured.*.{type : type, oper status : oper status}')
```

2. Json_query를 사용하여 Mtu가 1500인
인터페이스의 수를 구해주세요!
(!!MISSION!! 부분을 지우고 수정해주세요!)

```
- name: 9. GigabitEthernet이면서 up 상태이면서 mtu 1500인 인터페이스만 저장하기 (Save the in
  set_fact:
    interface_mtu: "{{ interface_gigetheretnet | json_query('[?!MISSION!?!]') }}"
```

##힌트

up인 상태의 인터페이스 수를 구하는 방법

```
json_query('[?oper_status==`up`]')
```



Thank you

