Ansible Network Automation Parsing everything in 2020

Ansiblefest 2020

Brad Thornton
Senior Principal Engineer

AnsibleFest

Agenda

What are we even talking about?

- Network configuration management
- Use cases for operational state data
- Introducing cli_parse
- Available parsing engines
- Advanced use cases
- Try it now

TL;DR

An Ansible strength

Configuration management

An Ansible weakness

Operational state assessment

Let's fix that.

[SOLVED] Configuration management

Network Resource Modules

- eos, ios, junos, nxos, vyos, xr
- acls, interfaces, l2_interfaces, l3_interfaces, lldp, ospf, etc

Protocol & platform modules

- eos, ios, junos, nxos, vyos, xr
- cli, netconf, http_api
- cli_config, netconf_config, eos_config

Operational state

the “show” commands

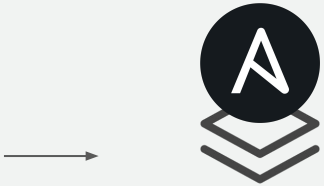
```
show interfaces

mgmt0 is up
admin state is up,
  Hardware: Ethernet, address: x200.0000.f8b5 (bia
abcd.0000.f8b5)
  Internet Address is 192.168.101.14/24

{"TABLE_interface": {"ROW_interface": [{"interface": "mgmt0",
"state": "up", "admin_state": "up", "eth_hw_desc":
"Ethernet", "eth_hw_addr": "x200.0000.f8b5", "eth_bia_addr":
"x200.0000.f8b5", "eth_ip_addr": "192.168.101.14",
"eth_ip_mask": "24", "eth_ip_prefix": "192.168.101.0", "et
h_mtu": "1500", "eth_bw": "1000000", "eth_dly": "10",
"eth_reliability":

<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply
xmlns="http://www.cisco.com/nxos:1.0:if_manager"
xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0">
 <nf:data>
  <show><interface>
```

Multiple formats



```
cli_command:
  register:
set_fact:
  | filter_plugin
  | lookup_plugin
```

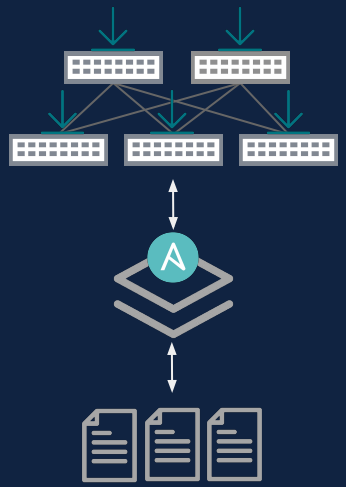
Multiple tasks and plugins

```
interfaces:
  admin:
    state:
      operating: up
      administrative: up
```

Desired format

Operational state data use cases

- Conditional task and roles within Ansible playbooks
  - Only make configuration changes if all bgp neighbors are healthy
- Fleet health assessment and inventory
  - Ensure all configured NTP servers are in sync
- Post change validation
  - LLDP, OSPF neighbors & reachability has not changed
- Custom reports using templates
  - Interface operating state vs. configured state



cli_parse

- New module available now
- Works with all platforms
- Work with many parsing engines
- Single task to run a command, parse & set facts
- Returns structured data from show command output

Show me the goods

```
tasks:
  - name: Run a command and parse results
    ansible.netcommon.cli_parse:
      command: show interfaces
      parser:
        name: ansible.netcommon.xxxx
      set_fact: interfaces
```

- Runs the command on the device
- Parse using the ‘xxxx’ engine
- Uses default template folder
- Parsed data set as fact
- Command output returned as stdout

Available parsing engines

- **ansible.netcommon.native**: Internal jinja, regex, yaml. No additional 3rd party libraries required
- **ansible.netcommon.ntc_templates**: Predefined textfsm templates packaged as python library
- **ansible.netcommon.pyats**: Cisco Test Automation & Validation Solution (11 OSs/2500 parsers)
- **ansible.netcommon.textfsm**: Python module for parsing semi-formatted text
- **ansible.netcommon.ttp**: Template based parsing, low regex use, jinja like DSL
- **ansible.netcommon.json:** convenience
- **ansible.netcommon.xml**: convert XML to json using xmltodict

Thank you library developers & contributors

Using the native parsing engine

```
---
- example: Ethernet1/1 is up
  getval: '(?P<name>\S+) is
(?P<oper_state>\S+)'
  result:
    "{{ name }}":
      name: "{{ name }}"
      state:
        operating: "{{ oper_state }}"
  shared: true

- example: admin state is up, Dedicated
Interface
  getval: 'admin state is (?P<admin_state>\S+)'
  result:
    "{{ name }}":
      state:
        admin: "{{ admin_state }}"

<...>
```



```
---
Ethernet1/1:
    hardware: 100/1000/10000
Ethernet
    mac_address: x200.005a.f8bd
    name: Ethernet1/1
    state:
      admin: up
      operating: up
Ethernet1/10:
    hardware: 100/1000/10000
Ethernet
    mac_address: x200.005a.f8c6
    name: Ethernet1/10
    state:
      admin: up
      operating: up
```

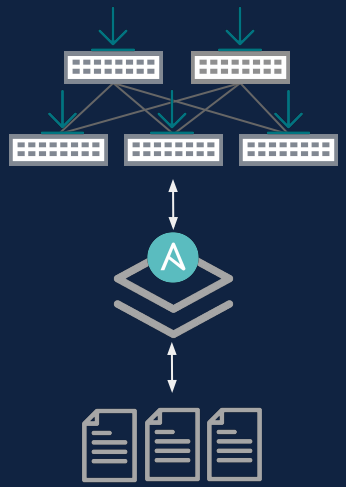
- Uses yaml templates
- Regular expression named capture groups
- Data built with jinja snippets
- Previous captures can be shared
- Used by network resource modules

Smart template discovery

```
templates/{{ os }}_{{ command }}.xyz
```

```
templates/eos_show_interfaces.yaml
```

```
templates/nxos_show_ntp_peers.textfsm
```



Advanced use cases for cli_parse

Override the default template name and path

```
- name: "Run command and parse with native"
  ansible.netcommon.cli_parse:
    command: show interface
    parser:
      name: ansible.netcommon.native
      template_path: /home/user/templates/filename.yaml
```

Command issued & parser command differ

```
- name: "Run command and parse with native"
  ansible.netcommon.cli_parse:
    command: sho int mgmt0
    parser:
      name: ansible.netcommon.native
      command: show interface
```

Override the default OS value

```
- name: Use ios instead of iosxe for pyats
  ansible.netcommon.cli_parse:
    command: show something
    parser:
      name: ansible.netcommon.pyats
      os: ios
```

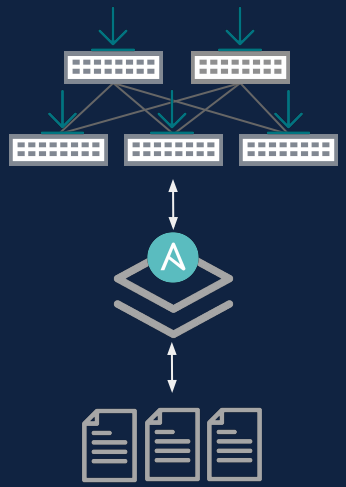
Parse text from file or previous task

```
- name: "Parse text from previous task"
  ansible.netcommon.cli_parse:
    text: "{{ sho_version['stdout'] }}"
    parser:
      name: ansible.netcommon.native
      command: show version

- name: "Parse text from file"
  ansible.netcommon.cli_parse:
    text: "{{ lookup('file', 'path/to/file.txt') }}"
    parser:
      name: ansible.netcommon.native
      template_path: /home/user/templates/filename.yaml
```

Works with linux too

```
hosts: fedora101
gather_facts: True
tasks:
- name: Use linux instead of fedora from ansible_distribution
  ansible.netcommon.cli_parse:
    command: ps -ef
    parser:
      name: ansible.netcommon.native
      os: linux
```



Developer Notes

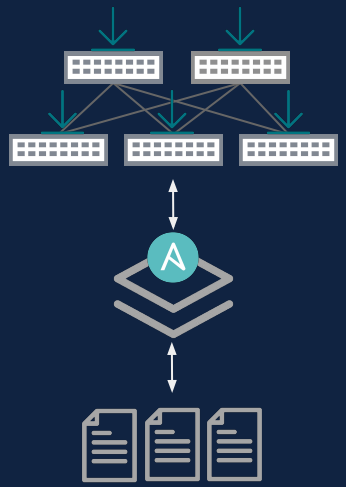
Under the hood

- **Plugin based architecture:** Loads parser plugins from collection plugin/cli_parsers directory
- **Simplified plugin requirements**:

```
class CliParser(CliParserBase):
    def parse(self, *_args, **kwargs):
```

- **Works with any collection:**

```
- name: Use a custom cli_parser
  ansible.netcommon.cli_parse:
    command: ls -l
    parser:
      name: my_organiztion.my_collection.custom_parser
```



Use it today

```
ansible-galaxy collection install ansible.netcommon
```

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

youtube.com/user/RedHatVideos

linkedin.com/company/Red-Hat

facebook.com/ansibleautomation

twitter.com/ansible