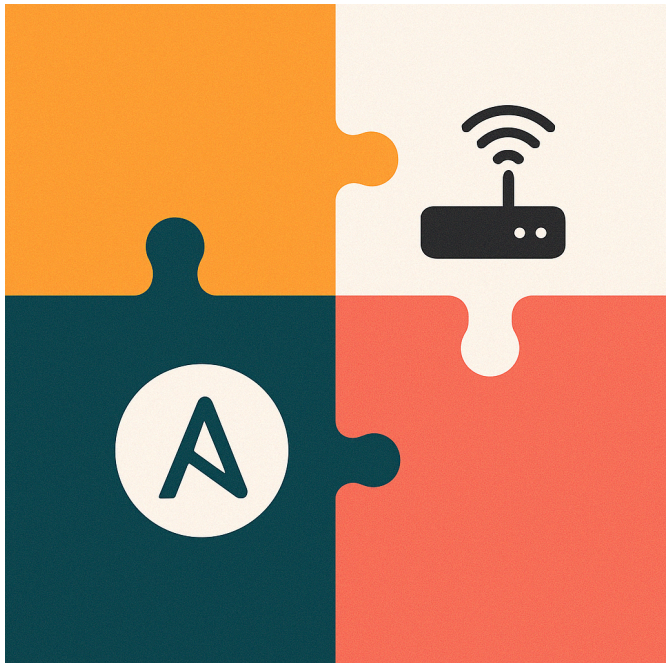
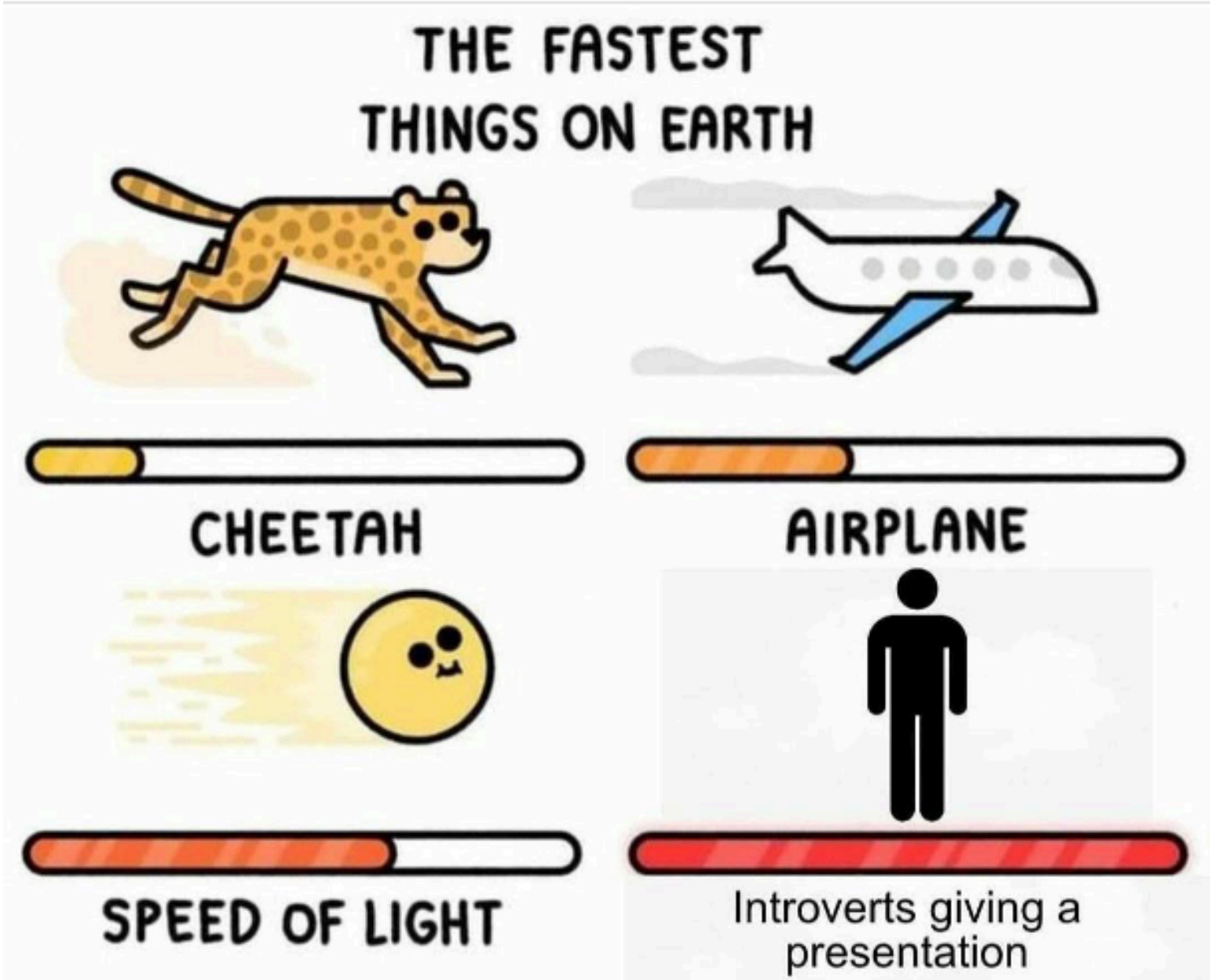


Agnostic automation with Network Resource Modules

Brandon Ewing

CHI-NOG 12, May 2025

www.github.com/bewing/chinog-12


THE FASTEST
THINGS ON EARTH
CHEETAH
AIRPLANE
SPEED OF LIGHT
Introverts giving a
presentation

Setup

- Brownfield Network
- No automated config management
- Multiple Vendors
- Desire to start partial config control
- Existing Ansible Infrastructure

Inventory

```yaml
network:
  vars:
    ansible_user: admin
    ansible_password: admin
  hosts:
    clab-chinog-ceos:
      ansible_network_os: arista.eos.eos
      ansible_connection: ansible.netcommon.httpapi
      ansible_httpapi_use_ssl: yes
      ansible_httpapi_validate_certs: no
    clab-chinog-iol:
      ansible_network_os: cisco.ios.ios
      ansible_connection: ansible.netcommon.network_cli
```

Take One

```
- hosts: all
  gather_facts: false
  tasks:
  - name: Set NTP servers
    ios_commands:
      commands:
      - configure terminal
      - ntp server 1.2.3.4
      - ntp server 4.3.2.1
```

Take One

```
- hosts: all
  gather_facts: false
  tasks:
  - name: Set NTP servers
    ios_commands:
      commands:
      - configure terminal
      - ntp server 1.2.3.4
      - ntp server 4.3.2.1
```

That works great!

```
$ ansible-playbook playbooks/take-one/take-one.yml -i inventory.yml -l clab-chinog-iol

PLAY [all] *********************************************************************

TASK [Set NTP servers] *********************************************************
ok: [clab-chinog-iol]

PLAY RECAP *********************************************************************
clab-chinog-iol : ok=1  changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
```

but...

Problems

- Change feedback?

```
changed=0
```

Problems

- Change feedback?

```
        changed=0
```

- What about removals?

```
ntp_servers:
- 4.3.2.1
- 9.8.7.6
```

Take Two

- Let's dig deeper

Take Two

- Let's dig deeper

- Manually get config/parse (TextFSM)
- Determine lines to add/remove (Jinja2 filters)
- Push changes to device (Jinja2 template)

Take Two

```yaml
- hosts: all
  gather_facts: no
  vars:
    ntp_servers:
    - 1.2.3.4
    - 4.3.2.1
  tasks:
  - name: Parse NTP config
    ansible.utils.cli_parse:
      command: show running-config
      parser:
        name: ansible.utils.textfsm
        template_path: "templates/fsm/ntp.fsm"
      set_fact:
        current_ntp_servers
  - name: Set NTP servers
    vars:
      to_add: "{{ ntp_servers | difference(current_ntp_servers | map(attribute='Host')) }}"
      to_delete: "{{ current_ntp_servers | map(attribute='Host') | difference(ntp_servers) }}"
    ios_config:
      src: "templates/config/ntp.j2"
```

Take Two

```
# ntp.fsm
Value Required Host (\S+)

Start
  ^ntp server $Host -> Record
```

```
{# ntp.j2 #}
{% for h in to_add %}
ntp server {{ h }}
{% endfor %}
{% for h in to_delete %}
no ntp server {{ h }}
{% endfor %}
```

That works great!

```
$ ansible-playbook playbooks/take-two/take-two.yml -i inventory.yml --limit clab-chinog-iol

PLAY [all] *********************************************************************

TASK [Parse NTP config] ********************************************************
ok: [clab-chinog-iol]

TASK [Debug] *******************************************************************
ok: [clab-chinog-iol] =>
    msg: []

TASK [Set NTP servers] *********************************************************
changed: [clab-chinog-iol]

PLAY RECAP *********************************************************************
clab-chinog-iol            : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

That works great!

```
$ ansible-playbook playbooks/take-two/take-two.yml -i inventory.yml --limit clab-chinog-iol

PLAY [all] *********************************************************************

TASK [Parse NTP config] ********************************************************
ok: [clab-chinog-iol]

TASK [Debug] *******************************************************************
ok: [clab-chinog-iol] =>
    msg: []

TASK [Set NTP servers] *********************************************************
changed: [clab-chinog-iol]

PLAY RECAP *********************************************************************
clab-chinog-iol            : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

but...

Problems

- Different OSes?

```
TASK [Set NTP servers] *********************************************************
fatal: [clab-chinog-ceos]: FAILED! =>
    changed: false
    msg: Connection type ansible.netcommon.httpapi is not valid for this module
ok: [clab-chinog-iol]

PLAY RECAP *********************************************************************
clab-chinog-ceos           : ok=2    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
clab-chinog-iol            : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Problems

- Different OSes?

```
TASK [Set NTP servers] *********************************************************
fatal: [clab-chinog-ceos]: FAILED! =>
    changed: false
    msg: Connection type ansible.netcommon.httpapi is not valid for this module
ok: [clab-chinog-iol]

PLAY RECAP *********************************************************************
clab-chinog-ceos           : ok=2    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
clab-chinog-iol            : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

- Complex configurations?
  - Do you authenticate your NTP?
  - How complex did your templates just get?

Complexity

```
{% if change_route_maps %}
{%  for route_map in change_route_maps %}
no route-map {{ route_map }}
{%   for index,term in target_route_maps[route_map].items()|sort %}
route-map {{ route_map }} {{ term['action'] }} {{ index }}
{%    for cmd in term['cmds'] %}
   {{ cmd }}
{%    endfor %}
{%   endfor %}
{%  endfor %}
{% endif %}
```

```
- name: Set route maps to change
  ansible.builtin.set_fact:
    change_route_maps: |-
      {%- set route_maps = {} %}
      {%- for route_map in current_route_maps.keys() | intersect(target_route_maps.keys()) %}
      {%-    set changes = [] %}
      {%-    for seq, current_details in current_route_maps[route_map].items() %}
      {%-       set target_details = target_route_maps[route_map].get(seq, None) %}
      {%-       if target_details is none %}
      {%-          set _ = changes.append("--route-map " ~ route_map ~ " permit " ~ seq) %}
      {%-          for cmd in current_details.cmds %}
      {%-             set _ = changes.append("--    " ~ cmd) %}
      {%-          endfor %}
      {%-          set _ = changes.append("--!") %}
      {%-       else %}
      {%-          for i in range(current_details.cmds | length) %}
      {%-             if current_details.cmds[i] != target_details.cmds[i] %}
      {%-                if i == 0 or current_details.cmds[i-1] == target_details.cmds[i-1] %}
      {%-                   set _ = changes.append("route-map " ~ route_map ~ " permit " ~ seq) %}
      {%-                endif %}
      {%-                set _ = changes.append("--    " ~ current_details.cmds[i]) %}
      {%-                set _ = changes.append("++    " ~ target_details.cmds[i]) %}
      {%-             endif %}
```

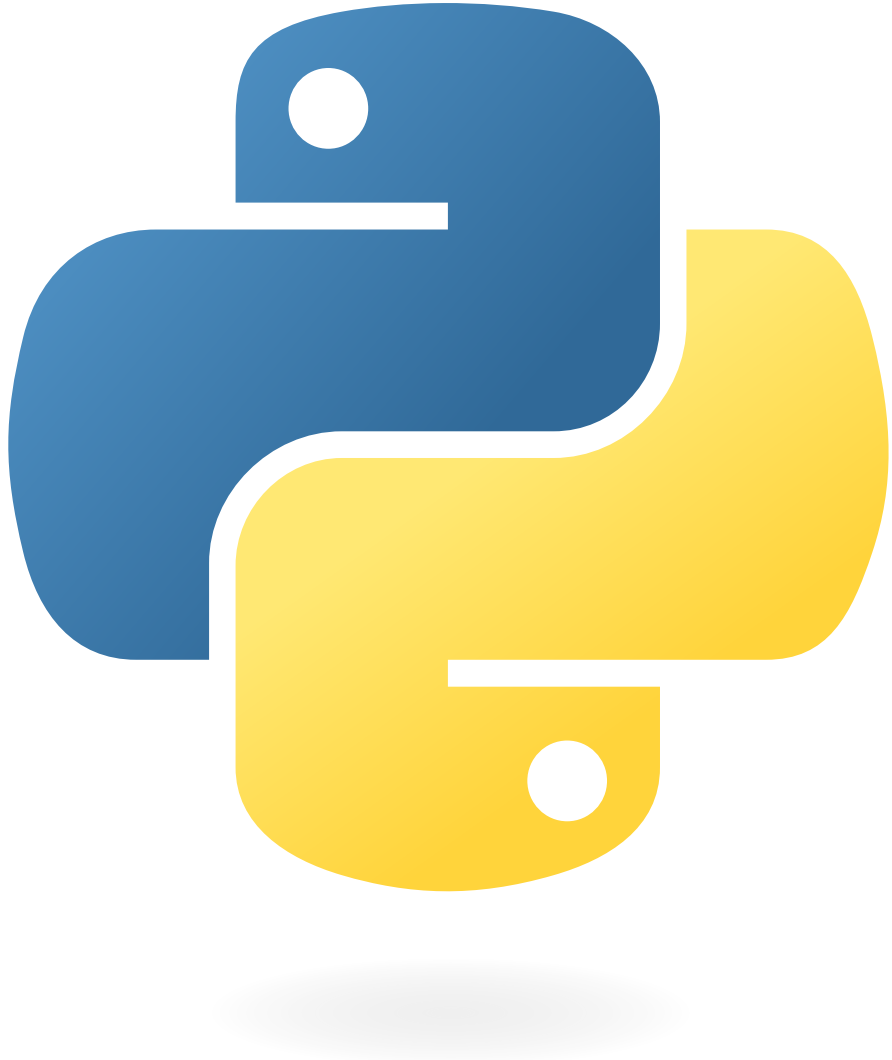
Complexity

Ansible, Jinja2, FSM, etc are Domain Specific Languages

Complexity

Ansible, Jinja2, FSM, etc are Domain Specific Languages

Let's move the complexity into a **programming language!**



Complexity

- Who's going to write all that code?!?!
  - Configuration modeling
  - CLI parsers
  - REST/RPC support?!?!

Complexity

- Who's going to write all that code?!?!
  - Configuration modeling
  - CLI parsers
  - REST/RPC support?!?!
- **AND** it all has to work within Ansible?

Complexity

- Who's going to write all that code?!?!
  - Configuration modeling
  - CLI parsers
  - REST/RPC support?!?!
- **AND** it all has to work within Ansible?

Red Hat Ansible team already did it!

Ansible Network Modules

- A set of vendor-specific Galaxy collections
- Written and supported by the Red Hat Ansible Network Team
  - with full documentation!
- Provide the networking framework
- Handles device fact collection (natively!) and config *section* management
- Supports check-mode execution

Ansible Network Modules

```yaml
- hosts: all
  gather_facts: false
  tasks:
  - name: Configure NTP
    register: return_value
    cisco.ios.ios_ntp_global:
      state: replaced
      config:
        authenticate: true
        authentication_keys:
        - algorithm: md5
          id: 2
          key: supersecret
          encryption: 7
        servers:
        - server: 1.2.3.4
          version: 2
          prefer: true
          key: 2
        - server: 4.3.2.1
          version: 2
          key: 2
  - ansible.builtin.debug:
      var: return_value
```

Ansible Network Modules

```
$ ansible-playbook playbooks/ios-ntp/ios-ntp.yml -i inventory.yml -l clab-chinog-iol

PLAY [all] *********************************************************************

TASK [Configure NTP] ***********************************************************
changed: [clab-chinog-iol]

TASK [ansible.builtin.debug] ***************************************************
ok: [clab-chinog-iol] =>
    mod_out:
        after:
            authentication_keys:
            -   algorithm: md5
                encryption: 7
                id: 2
                key: VALUE_SPECIFIED_IN_NO_LOG_PARAMETER
            servers:
            -   key_id: 2
                prefer: true
                server: 1.2.3.4
                version: 2
            -   key_id: 2
                server: 4.3.2.1
                version: 2
        before: {}
        changed: true
        commands:
        - ntp authentication-key 2 md5 ******** 7
        - ntp server 1.2.3.4 key 2 prefer version 2
        - ntp server 4.3.2.1 key 2 version 2
        failed: false

PLAY RECAP *********************************************************************
clab-chinog-iol            : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Ansible Network Modules

```
$ ansible-playbook playbooks/ios-ntp/ios-ntp.yml -i inventory.yml -l clab-chinog-iol

PLAY [all] *********************************************************************

TASK [Configure NTP] ***********************************************************
ok: [clab-chinog-iol]

TASK [ansible.builtin.debug] ***************************************************
ok: [clab-chinog-iol] =>
    mod_out:
        before:
            authentication_keys:
            -   algorithm: md5
                encryption: 7
                id: 2
                key: VALUE_SPECIFIED_IN_NO_LOG_PARAMETER
            servers:
            -   key_id: 2
                prefer: true
                server: 1.2.3.4
                version: 2
            -   key_id: 2
                server: 4.3.2.1
                version: 2
        changed: false

PLAY RECAP *********************************************************************
clab-chinog-iol            : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

So what's actually going on

Ansible Network Modules contain:

- An *argspec* covering the model of the config (usually the **want** part)
- A *facts* module that handles populating the **have** argspec from the device
- A *config* module that can compare want/have and generate a list of commands

argspec

Arguments passed to the module.

- the **state** directive
  - Controls module behavior

argspec

Arguments passed to the module.

- the **state** directive
  - Controls module behavior
- **config** dictionary contains the model for this module
  - *similar* between OS/collections but NOT IDENTICAL

argspec

Arguments passed to the module.

- the **state** directive
  - Controls module behavior
- **config** dictionary contains the model for this module
  - *similar* between OS/collections but NOT IDENTICAL
- **EXTREMELY** well-documented, with extensive examples

argspec

states

- merged
- replaced
- overridden
- deleted
- gathered
- rendered

argspec

state: merged

- Usually the safest
- Will ADD your config to the existing config
- and keep what is already present
- Will merge down into individual ACL entries
- So if you have seq 10 20 30, and merge in 15 and 20
  - 15 is added
  - 20 is replaced

merged

replaced

overridden

deleted

gathered

rendered

argspec

state: replaced

Replace a subsection (usually) of the config section on the device

- Used to replace the configured subsection with the provided config
- Differs per module
  - NTP will behave like `overidden`
  - But ACL may only replace a single ACL
- Please read the docs and perform testing when using this in plays/roles

merged

replaced

overridden

deleted

gathered

rendered

argspec

state: overridden

Override the entire section on the device

- In some cases, same behavior as `replaced` (NTP, etc)
- In others, may remove config! (ACL, prefix list, etc)
- As usual, test when possible

merged

replaced

overridden

deleted

gathered

rendered

argspec

state: deleted

merged

replaced

overridden

deleted

gathered

rendered

argspec

state: deleted

REMOVES THE SECTION FROM THE DEVICES

use with care

merged

replaced

overridden

deleted

gathered

rendered

state: gathered

- Special state, only gathers **have**
  - returns as `gathered`

```
$ ansible-playbook playbooks/gather/gather.yml -i inventory.yml -l clab-chinog-iol
TASK [Gather NTP config] ********************************************************
ok: [clab-chinog-iol]

TASK [ansible.builtin.debug] ****************************************************
ok: [clab-chinog-iol] =>
    output:
        changed: false
        failed: false
        gathered:
            authentication_keys:
            -   algorithm: md5
                encryption: 7
                id: 2
                key: VALUE_SPECIFIED_IN_NO_LOG_PARAMETER
            servers:
            -   key_id: 2
                prefer: true
                server: 1.2.3.4
                version: 2
            -   key_id: 2
                server: 4.3.2.1
                version: 2
```

merged

replaced

overridden

deleted

gathered

rendered

state: rendered

Inverts `gathered`, takes your `argspec.config`/**want** and turns it into a configuration

NB: Does *NOT* connect to device / collect config

```
TASK [ansible.builtin.debug] ***************************************************
ok: [clab-chinog-iol] =>
    output:
        changed: false
        failed: false
        rendered:
        - ntp authenticate
        - ntp authentication-key 2 md5 ******** 7
        - ntp server 1.2.3.4 key 2 prefer version 2
        - ntp server 4.3.2.1 key 2 version 2
```

merged

replaced

overridden

deleted

gathered

rendered

- Well documented
- Examples for *EVERY* state in every module
- Similar, but not *identical* config specs between vendors
- Highly advise testing prior to deployment

- Well documented
- Examples for *EVERY* state in every module
- Similar, but not *identical* config specs between vendors
- Highly advise testing prior to deployment

- See playbooks/states in Github for an example with the IOS ACL module

facts

gather_facts: false

facts

gather_facts: ~~false~~

gather_facts: it_depends

facts

Device fact modules

- You can call individual OS fact modules as tasks

```
- hosts: all
  gather_facts: false
  tasks:
  - cisco.ios.ios_facts:
      gather_subset:
      - interfaces
      gather_network_resources:
      - ntp_global
  - ansible.builtin.debug:
      var: ansible_facts.keys()
  - ansible.builtin.debug:
      var: ansible_facts.network_resources.keys()
```

facts

```
$ ansible-playbook playbooks/facts/facts.yml -i inventory.yml -l clab-chinog-iol

PLAY [all] *********************************************************************

TASK [cisco.ios.ios_facts] *****************************************************
ok: [clab-chinog-iol]

TASK [ansible.builtin.debug] ***************************************************
ok: [clab-chinog-iol] =>
    ansible_facts.keys():
    - network_resources
    - net_gather_network_resources
    - net_gather_subset
    - net_all_ipv4_addresses
    - net_all_ipv6_addresses
    - net_neighbors
    - net_interfaces
    - net_system
    - net_image
    - net_version
    - net_hostname
    - net_api
    - net_python_version
    - net_iostype
    - net_operatingmode
    - net_serialnum

TASK [ansible.builtin.debug] ***************************************************
ok: [clab-chinog-iol] =>
    ansible_facts.network_resources.keys():
    - ntp_global

PLAY RECAP *********************************************************************
clab-chinog-iol            : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

facts

- `gather_facts` is a play-level keyword
- so is `gather_subset`!

facts

- `gather_facts` is a play-level keyword
- so is `gather_subset`!

- `gather_network_resources` is not :(

```
ERROR! 'gather_network_resources' is not a valid attribute for a Play
```

facts

HOWEVER

`gather_facts` at the play-level is just calling `ansible.builtin.gather_facts`

facts

HOWEVER

`gather_facts` at the play-level is just calling `ansible.builtin.gather_facts`

`module_defaults` to the rescue!

```
- hosts: all
  module_defaults:
    ansible_builtin.gather_facts:
      gather_subset:
      - interfaces
      gather_network_resources:
      - ntp_global
  gather_facts: true
  tasks:
  - ansible.builtin.debug:
    var: ansible_facts
```

```
TASK [ansible.builtin.debug] ***************************************************
ok: [clab-chinog-iol] =>
    ansible_facts:
        net_all_ipv4_addresses:
        - 172.20.20.3
        net_all_ipv6_addresses:
        - 3FFF:172:20:20::3
```

Problems

- Different OSes?

```
TASK [Set NTP servers] *********************************************************
TASK [Set NTP servers] *********************************************************
fatal: [clab-chinog-ceos]: FAILED! =>
    changed: false
    msg: Connection type ansible.netcommon.httpapi is not valid for this module
ok: [clab-chinog-iol]

PLAY RECAP *********************************************************************
clab-chinog-ceos           : ok=2    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
clab-chinog-iol            : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Interoperability

```
- hosts: all
  gather_facts: false
  tasks:
  - name: Configure NTP
    cisco.ios.ios_ntp_global:
      state: replaced
      config:
      ...
```

Interoperability

```
- hosts: all
  gather_facts: false
  tasks:
  - name: Configure NTP
    cisco.ios.ios_ntp_global:
      state: replaced
      config:
      ...
```

```
- hosts: all
  gather_facts: false
  tasks:
  - name: Configure NTP
    ansible.netcommon.network_resource:
      name: ntp_global
      state: replaced
      config:
      ...
```

Interoperability

```
$ ansible-playbook playbooks/agnostic-ntp/agnostic-ntp.yml -i inventory.yml

PLAY [all] *********************************************************************

TASK [Configure NTP] ***********************************************************
changed: [clab-chinog-ceos]
ok: [clab-chinog-iol]

TASK [ansible.builtin.debug] ***************************************************
ok: [clab-chinog-ceos] =>
    mod_out:
        after:
            authentication_keys:
            -   algorithm: md5
                encryption: 7
                id: 2
                key: VALUE_SPECIFIED_IN_NO_LOG_PARAMETER
            servers:
            -   key_id: 2
                prefer: true
                server: 1.2.3.4
                version: 2
            -   key_id: 2
                server: 4.3.2.1
                version: 2
        ansible_connection: ansible.netcommon.httpapi
        ansible_network_os: arista.eos.eos
<snip>
PLAY RECAP *********************************************************************
clab-chinog-ceos           : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
clab-chinog-iol            : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Interoperability

That works great!

but...

argspec

Arguments passed to the module.

- the **state** directive
  - Controls module behavior
- **config** dictionary contains the model for this module
  - ***similar* between OS/collections but NOT IDENTICAL**

Interoperability

arista.eos.ntp_global

local_interface string	Configure the interface from which the IP source address is taken.

cisco.ios.ntp_global

source string	Configure interface for source address

Interoperability

ansible-network/resource_module_modules#298

Interoperability

```
- name: Configure Cisco NTP
  when: ansible_network_os == "cisco.ios.ios"
  ansible.netcommon.network_resource:
    name: ntp_global
    config:
      local_interface: Loopback0
...
- name: Configure Arista NTP
  when: ansible_network_os == "arista.eos.eos"
  ansible.netcommon.network_resource:
    name: ntp_global
    config:
      source: Loopback0
```

Interoperability

HostVars

```
network:
  hosts:
    clab-chinog-ceos:
      ntp_config:
        local_interface: Ethernet1
        servers:
        - server: 1.2.3.4
          server: 4.3.2.1
```

```
tasks:
- name: Push NTP config
  ansible.netcommon.network_resource
    name: ntp_global
    state: overridden
    config: "{{ ntp_config }}""
```

Interoperability

- Resource modules commit, but do not save!
- Define handlers in your plays to write startup-config if device has one

```
handlers:
- name: Save IOS config
  listen: Save config
  when: not ansible_check_mode and ansible_network_os == "cisco.ios.ios"
  ansible.netcommon.cli_command:
     command: copy running-config startup-config
     prompt: Destination
     answer: '\r'

- name: Save EOS config
  listen: Save config
  when: not ansible_check_mode and ansible_network_os == "arista.eos.eos"
  arista.eos.eos_command: # cli_command doesn't support httpapi
    commands:
    - copy running-config startup-config
```

ansible-collections/ansible.netcommon#692

Demo

https://youtu.be/uRkVllY1FaY

imc

Questions?

CHI-NOG 12, May 2025
www.github.com/bewing/chinog-12