Ansible Network Automation

Live Training Session
December 2021

Day2 Schedule - Network Fact Gathering & Templating

platform_command Modules (basics)
Using cli_command
Directly Passing Credentials /Privilege Escalation
Jinja2 as part of Ansible
Configuration Templating Basics
Advanced Configuration Templating
Block/Rescue/Always
Network Configuration Overview
Resource Modules

Network Devices: Basic Show Operations

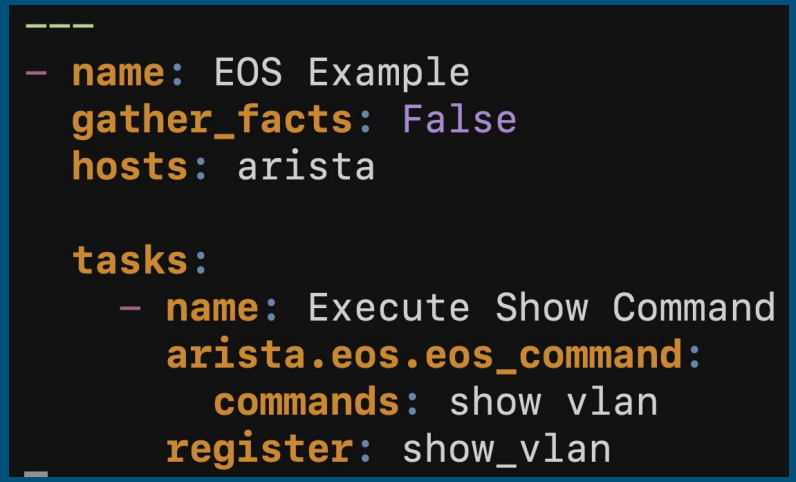

```yaml
---
- name: IOS Example
  hosts: cisco
  gather_facts: false
  tasks:
    - name: "Test *_facts"
      cisco.ios.ios_facts:
      register: cisco_facts
```

```yaml
---
- name: Arista Example
  hosts: arista
  gather_facts: false
  tasks:
    - arista.eos.eos_facts:

    - ansible.builtin.debug:
        var: ansible_net_model

    - ansible.builtin.debug:
        msg:
          - "{{ ansible_net_system }}"
          - "{{ ansible_net_version }}"
```

Reference Material in:

{{ github_repo }}/platform_command

Network Devices: Basic Show Operations



```yaml
---
- name: EOS Example
  gather_facts: False
  hosts: arista

  tasks:
    - name: Execute Show Command
      arista.eos.eos_command:
        commands: show vlan
      register: show_vlan
```

Module (FQCN)

Network Devices: Basic Show Operations (where is my JSON)


```yaml
---
- name: EOS Example
  gather_facts: False
  hosts: arista

  tasks:
    - name: Show command using pipe json
      arista.eos.eos_command:
        commands: show interfaces | json
      register: output

    - ansible.builtin.debug:
        var: output.stdout[0]["interfaces"]
```

Network Devices: Basic Show Operations (list of commands)


```
---
- name: NX-OS Example
  gather_facts: False
  hosts: nxos

  tasks:
    - name: "Show command (list of commands)"
      cisco.nxos.nxos_command:
        commands:
          - show ip int brief vrf management
          - show ip arp vrf management
      register: output

    - ansible.builtin.debug:
        var: output
```

Network Devices: Basic Show Operations (prompting)


```yaml
- name: IOS Prompting Example
  hosts: cisco5:cisco6
  gather_facts: False
  tasks:
    - cisco.ios.ios_command:
        commands:
          - command: clear counters gigabitEthernet1
            # Regular expression for "prompt"--simplify
            # Clear "show interface" counters on this interface [confirm]
            prompt: confirm
            answer: "y"
      register: output
```

Network Devices: More Generic (cli_command)


```yaml
---
- name: cli_command Example
  hosts: cisco:arista
  gather_facts: False
  tasks:
    - ansible.netcommon.cli_command:
        command: show ip int brief
      register: output

    - debug:
        msg: "{{ output }}"
```

```
[arista:vars]
ansible_network_os=eos
```

Exercises:
./day2/platform_command/ex1.txt
./day2/platform_command/ex2.txt
./day2/platform_command/ex3.txt
./day2/platform_command/ex4.txt
./day2/platform_command/ex5.txt

Directly Passing Credentials

```
ansible_user=pyclass
ansible_ssh_pass=my_password
```

Nope, no thanks.

```
$ ansible-playbook example.yml -i ./ansible-hosts.ini -u pyclass -k
SSH password:
```

Exercises:
./day2/priv_escalation/ex1.txt
./day2/priv_escalation/ex2.txt

Privilege Escalation (where is my enable secret)

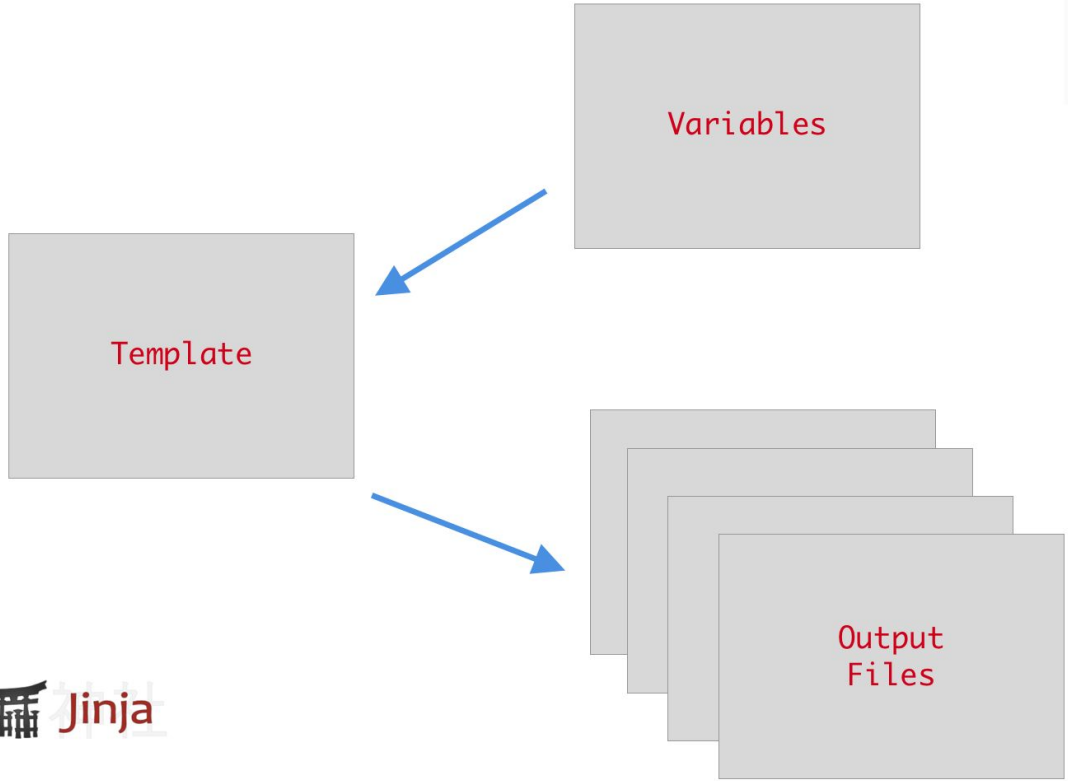
```yaml
- name: Enable Example
  hosts: cisco1
  gather_facts: False
  become: true
  become_method: enable

  tasks:
    - name: Execute show run
      cisco.ios.ios_command:
        commands: show run
      register: output
```

Become

Args

```
$ ansible-playbook example2.yml -i ./ansible-hosts.ini -u test9 -k -K
SSH password:
BECOME password[defaults to SSH password]:
```

Jinja2 Pattern



```
Variables: "{{ my_var }}"


Jinja2 Filters: "{{ result.stdout | from_json }}"


Template Lookup:
- debug:
    msg: "{{ lookup('template', './some_template.j2') }}"


Template Module:
- name: Generate Configuration Files
  template:
    src: router.j2
    dest: /home/gituser/ANSIBLE/CFGS/{{ hostname }}.txt
```

router.j2

```
no service pad
service tcp-keepalives-in
service tcp-keepalives-out
service timestamps debug datetime msec
localtime show-timezone
service timestamps log datetime msec
localtime show-timezone
service password-encryption
!
hostname {{ inventory_hostname }}
!
boot-start-marker
boot-end-marker
!
logging buffered 32000
no logging console
```

Jinja

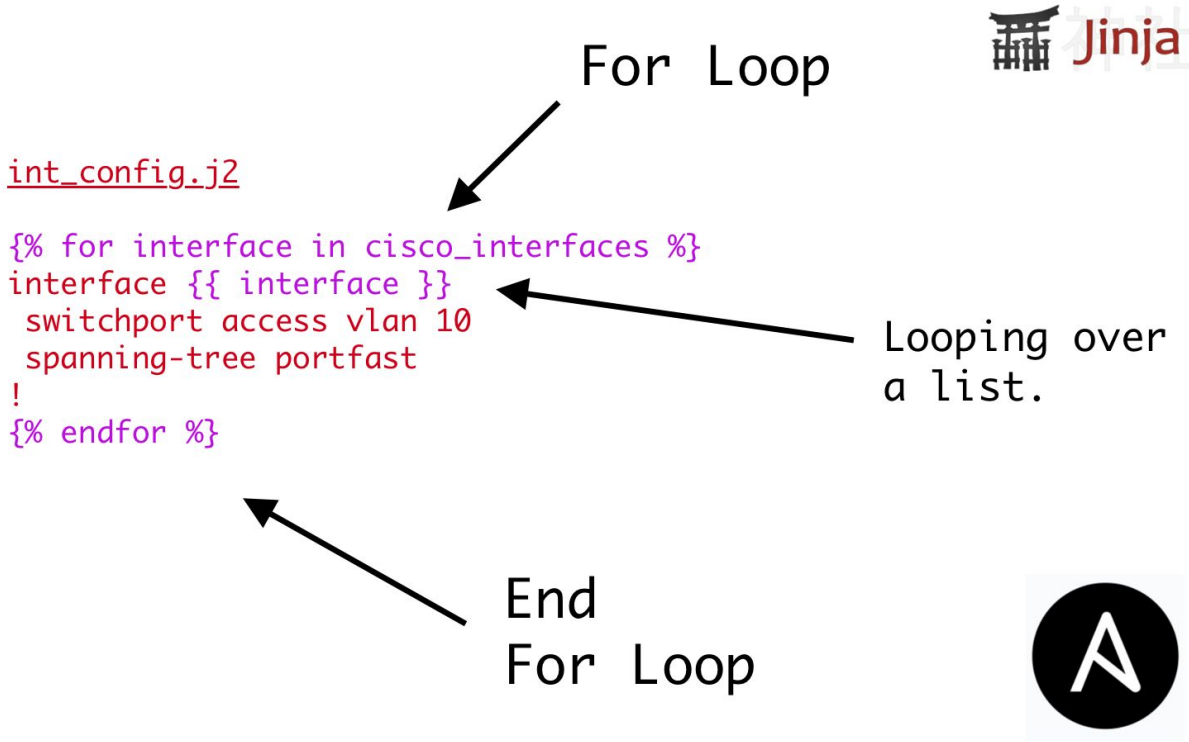
Variable

dhcp_config.j2

Conditional

```
{% if dhcp %}
no ip dhcp conflict logging
ip dhcp excluded-address {{ dhcp_excl_start }} {{ dhcp_excl_end }}
!
ip dhcp pool POOL1
   network {{ dhcp_network }} {{ dhcp_netmask }}
   default-router {{ dhcp_gateway }}
   dns-server 8.8.8.8 8.8.4.4
{% endif %}
```

End Conditional


For Loop
Jinja
int_config.j2
{% for interface in cisco_interfaces %}
interface {{ interface }}
switchport access vlan 10
spanning-tree portfast
!
{% endfor %}
Looping over a list.
End For Loop

Looping over
a dictionary

int_config2.j2

```
{% for port_name, port_attribs in ports.items() %}
interface {{ port_name }}
{% if port_attribs.mode == 'access' %}
 switchport mode access
 switchport access vlan {{ port_attribs.access_vlan }}
{% endif %}
!
{% endfor %}
```

End
For Loop

Nesting templates with include

router_includes.j2

```
!
hostname {{ inventory_hostname }}
!
boot-start-marker
boot-end-marker
!
logging buffered 32000
no logging console
!
{% include '881_interfaces.j2' %}
!
```

Pulling in other templates

modular_template.j2

```
!
{% include "snmp.j2" %}
!
!
!
control-plane
!
!
...
!
scheduler max-task-time 5000
{% include "ntp.j2" %}
!
end
```

Integrating Jinja2 with Ansible

```
- name: Test Jinja2
  hosts: cisco
  gather_facts: False
  tasks:
    - ansible.builtin.template:
        src: router.j2
        dest: "./CFGS/{{ inventory_hostname }}.txt"
```

Reference Material in:
{{ github_repo }}/jinja2

Modular Jinja2 Templates

```
{% include "services.j2" %}
!
hostname cisco1
!
boot-start-marker
boot system flash c880data-universalk9-mz.155-3.M8.bin
boot system flash c880data-universalk9-mz.154-2.T1.bin
boot-end-marker
!
!
logging buffered 10000
no logging console
enable secret 5 $1$39ck$mpfZ3sWVn10YCXhSWI48R/
!
{% include "aaa.j2" %}
ethernet lmi ce
memory-size iomem 10
clock timezone PST -8 0
clock summer-time PDT recurring
```

Reference Material in:

{{ github_repo }}/jinja2_modular

Exercises:

./day2/jinja2/ex1.txt

./day2/jinja2/ex2.txt

Block / Rescue / Always

```
- name: Block Example
  hosts: cisco:arista:juniper
  gather_facts: False
  tasks:
    - block:
        - name: Executing command
          cisco.ios.ios_command:
            commands: show ip int brief
          register: output

        - ansible.builtin.set_fact:
            show_ip: "{{ output.stdout[0] }}"

        - ansible.builtin.debug:
            msg: "{{ show_ip.splitlines() }}"
      when: 'ansible_network_os == "ios"'
```

Reference Material in:

{{ github_repo }}/block_examples

Ansible Configuration: A Maze of Options

General Solutions vs Specific Solutions

General Solutions
*_config (ios_config, eos_config, et cetera)
napalm-ansible

Specific Solutions
Resource Modules
net_modules
Feature specific modules

PYTHON
FOR NETWORK ENGINEERS

Ansible is Fundamentally about Config Management.

Ansible Configuration:

* Idempotency matters

* Declarative

* ACID-like Transactions

PYTHON
FOR NETWORK ENGINEERS

Declarative Configuration:

When we state—what exactly are we specifying?

```
eos_vlans:
  config:
    - name: Blue10
      vlan_id: 10
      state: active
```

What happens to other VLANs besides VLAN 10? Do they get removed?

What if we drop the name? Does the old name get removed (set back to the default value)

Is this incremental additional or the entire set of VLANS?

Networking Modules

Modules

- nxos_aaa_server – Manages AAA server global configuration.
- nxos_aaa_server_host – Manages AAA server host-specific configuration.
- nxos_acl – (deprecated, removed after 2022-06-01) Manages access list entries for ACLs.
- nxos_acl_interface – (deprecated, removed after 2022-06-01) Manages applying ACLs to interfaces.
- nxos_acl_interfaces – ACL interfaces resource module
- nxos_acls – ACLs resource module
- nxos_banner – Manage multiline banners on Cisco NXOS devices
- nxos_bfd_global – Bidirectional Forwarding Detection (BFD) global-level configuration
- nxos_bfd_interfaces – BFD interfaces resource module
- nxos_bgp – (deprecated, removed after 2023-01-27) Manages BGP configuration.
- nxos_bgp_address_family – BGP Address Family resource module.
- nxos_bgp_af – (deprecated, removed after 2023-02-24) Manages BGP Address-family configuration.
- nxos_bgp_global – BGP Global resource module.
- nxos_bgp_neighbor – (deprecated, removed after 2023-01-27) Manages BGP neighbors configurations.
- nxos_bgp_neighbor_address_family – BGP Neighbor Address Family resource module.
- nxos_bgp_neighbor_af – (deprecated, removed after 2023-02-24) Manages BGP address-family's neighbors configuration.
- nxos_command – Run arbitrary command on Cisco NXOS devices
- nxos_config – Manage Cisco NXOS configuration sections
- nxos_devicealias – Configuration of device alias for Cisco NXOS MDS Switches.
- nxos_evpn_global – Handles the EVPN control plane for VXLAN.
- nxos_evpn_vni – Manages Cisco EVPN VXLAN Network Identifier (VNI).
- nxos_facts – Gets facts about NX-OS switches
- nxos_feature – Manage features in NX-OS switches.
- nxos_file_copy – Copy a file to a remote NXOS device.
- nxos_gir – Trigger a graceful removal or insertion (GIR) of the switch.

Modules, modules, and more modules

Ansible Resource Modules

Resource Modules

There is a deprecation of some of feature specific modules that came before resource modules.

There is an implied data model that is associated with the resources module. This same data model is also associated with the facts for that resource.

Ansible Resource Modules

Coupling of a data model and configuration.

You can retrieve information from the device in the format of the implied data model.

arista.eos

- arista.eos.eos_acl_interfaces – ACL interfaces resource module
- arista.eos.eos_acls – ACLs resource module
- arista.eos.eos_banner – Manage multiline banners on Arista EOS devices
- arista.eos.eos_bgp – (deprecated, removed after 2023-01-29) Configure global BGP protocol settings on Arista EOS.
- arista.eos.eos_bgp_address_family – Manages BGP address family resource module
- arista.eos.eos_bgp_global – Manages BGP global resource module
- arista.eos.eos_command – Run arbitrary commands on an Arista EOS device
- arista.eos.eos_config – Manage Arista EOS configuration sections
- arista.eos.eos_eapi – Manage and configure Arista EOS eAPI.
- arista.eos.eos_facts – Collect facts from remote devices running Arista EOS
- arista.eos.eos_interface – (deprecated, removed after 2022-06-01) Manage Interface on Arista EOS network devices
- arista.eos.eos_interfaces – Interfaces resource module
- arista.eos.eos_l2_interface – (deprecated, removed after 2022-06-01) Manage L2 interfaces on Arista EOS network devices.
- arista.eos.eos_l2_interfaces – L2 interfaces resource module
- arista.eos.eos_l3_interface – (deprecated, removed after 2022-06-01) Manage L3 interfaces on Arista EOS network devices.
- arista.eos.eos_l3_interfaces – L3 interfaces resource module
- arista.eos.eos_lacp – LACP resource module
- arista.eos.eos_lacp_interfaces – LACP interfaces resource module
- arista.eos.eos_lag_interfaces – LAG interfaces resource module
- arista.eos.eos_linkagg – (deprecated, removed after 2022-06-01) Manage link aggregation groups on Arista EOS network devices
- arista.eos.eos_lldp – Manage LLDP configuration on Arista EOS network devices
- arista.eos.eos_lldp_global – LLDP resource module
- arista.eos.eos_lldp_interfaces – LLDP interfaces resource module
- arista.eos.eos_logging – Manage logging on network devices
- arista.eos.eos_logging_global – Manages logging resource module
- arista.eos.eos_ntp_global – Manages ntp resource module
- arista.eos.eos_ospf_interfaces – OSPF Interfaces Resource Module.
- arista.eos.eos_ospfv2 – OSPFv2 resource module
- arista.eos.eos_ospfv3 – OSPFv3 resource module
- arista.eos.eos_prefix_lists – Manages Prefix lists resource module
- arista.eos.eos_route_maps – Manages Route Maps resource module
- arista.eos.eos_static_route – (deprecated, removed after 2022-06-01) Manage static IP routes on Arista EOS network devices
- arista.eos.eos_static_routes – Static routes resource module

What is the data-model?

```
ok: [arista5] => {
    "ansible_network_resources.vlans": [
        {
            "state": "active",
            "vlan_id": 2
        },
        {
            "state": "active",
            "vlan_id": 3
        },
        {
            "state": "active",
            "vlan_id": 4
        },
```

Reference Material in:

{{ github_repo }}/resource_modules_facts

```
- name: RM Facts
  hosts: arista5
  gather_facts: True
  module_defaults:
    eos_facts:
      gather_network_resources:
        - vlans
        - interfaces
        - l2_interfaces
        - l3_interfaces
        - lacp
        - lldp_global
        - lldp_interfaces
        - lag_interfaces

  tasks:
    - name: Test
      ansible.builtin.debug:
        var: ansible_network_resources
      tags: rm_all
```

What is the data-model?

```
- name: VLAN configuration (gathered)
  hosts: arista5
  gather_facts: False
  tasks:
    - name: Test gathering
      arista.eos.eos_vlans:
        state: gathered
      register: vlans

    - ansible.builtin.debug:
        var: vlans.gathered
```

Resource Modules: States

https://docs.ansible.com/ansible/latest/network/user_guide/network_resource_modules.html

Network resource module states

You use the network resource modules by assigning a state to what you want the module to do. The resource modules support the following states:

merged

Ansible merges the on-device configuration with the provided configuration in the task.

replaced

Ansible replaces the on-device configuration subsection with the provided configuration subsection in the task.

overridden

Ansible overrides the on-device configuration for the resource with the provided configuration in the task. Use caution with this state as you could remove your access to the device (for example, by overriding the management interface configuration).

deleted

Ansible deletes the on-device configuration subsection and restores any default settings.

gathered

Ansible displays the resource details gathered from the network device and accessed with the `gathered` key in the result.

rendered

Ansible renders the provided configuration in the task in the device-native format (for example, Cisco IOS CLI). Ansible returns this rendered configuration in the `rendered` key in the result. Note this state does not communicate with the network device and can be used offline.

parsed

Ansible parses the configuration from the `running_config` option into Ansible structured data in the `parsed` key in the result. Note this does not gather the configuration from the network device so this state can be used offline.

Resource Modules: States

Merged - Configure the things you specify and nothing else.

Replaced - For the given section of the config, replace that section.

Overridden - For the given resource, completely replace that resource with the provided configuration.

Using Resource Modules

Reference Material in:
{{ github_repo }}/resource_modules

```
- name: VLAN configuration
  hosts: arista5
  gather_facts: False
  tasks:
    - name: Merge provided configuration with device configuration
      arista.eos.eos_vlans:
        config:
          - name: Blue10
            vlan_id: 10
            state: active
          - name: Blue20
            vlan_id: 20
            state: active
          - name: Blue30
            vlan_id: 30
            state: active
        state: merged
```

```
$ ansible-playbook vlans_eos_merged.yml --check -vvv
```

```
],
"changed": true,
"commands": [
    "vlan 2",
    "name VLAN0002",
    "vlan 3",
    "name VLAN0003",
    "vlan 4",
    "name VLAN0004",
    "vlan 5",
    "name VLAN0005",
    "vlan 6",
    "name VLAN0006",
    "vlan 7",
    "name VLAN0007",
    "no vlan 10",
    "no vlan 20",
    "no vlan 30",
    "vlan 1",
    "name default",
    "state active"
],
"invocation": {
```

Find the commands section in the output. These are the commands that will be configured on the device.

Exercises:
./day2/resource_modules/ex1.txt
./day2/resource_modules/ex2.txt