Ansible Automation Stack: Dependencies, Environment, and Design Principles

Dependencies & Environment

To ensure the automation stack works properly, the following dependencies must be installed on the Ansible control node:

System Requirements

- **Python 3.10+**
- **Ansible 2.12+ (recommended: 2.17.x)**
- OS: Tested on Ubuntu 22.04+

Python & Ansible Packages

Install with pip or use a virtualenv:

```
pip install ansible==2.17.10
pip install jinja2<3.2  # avoid compatibility issues
pip install ncclient    # required for Junos NETCONF
```

Ansible Collections

Install required Ansible collections:

```
ansible-galaxy collection install cisco.ios
ansible-galaxy collection install junipernetworks.junos
ansible-galaxy collection install ansible.netcommon
```

Optional: `paramiko` is used as fallback SSH transport if `ansible-pylibssh` is not installed.

Design Architecture

Directory Layout

```
ansible/
├── ansible.cfg
├── hosts.yml
├── group_vars/
│   ├── cisco.yml
│   ├── juniper.yml
│   └── ruijie.yml
├── playbooks/
│   ├── port/
│   │   ├── toggle_port.yml
│   │   └── tasks/
│   │       ├── enable
│   │       |   ├── ios_enable.yml
│   │       |   └── junos_enable.yml
│   │       └── disable
│   │           ├── ios_disable.yml
│   │           └── junos_disable.yml
│   ├── vlan/
│   │   ├── get_vlan_info.yml
│   │   ├── assign_vlan.yml
│   │   └── tasks/
│   │       ├── assing_vlan
│   │       │   ├── ios_assign_vlan.yml
│   │       │   └── junos_assign_vlan.yml
│   │       └── get_vlan
│   │           ├── cisco_get_vlan.yml
│   │           ├── juniper_get_vlan.yml
│   │           └── ruijie_get_vlan.yml
│   ├── interfaces/
│   │   ├── get_interfaces.yml
│   │   └── tasks/
│   │       ├── ios_show_interfaces.yml
│   │       └── junos_show_interfaces.yml
│   └── coa/
│       ├── send_coa.yml
│       └── templates/
│           ├── coa_mac.tpl
│           └── coa_user.tpl
```

Key Design Principles

1. Vendor-Based Routing

- Devices are grouped in inventory (e.g. `cisco` , `juniper` , `ruijie`)
- Each group defines a `vendor` variable in `group_vars/`
- Playbooks use `include_tasks: tasks/{{ vendor }}_whatever.yml` to branch logic

2. Connection Control

- `ansible_network_os` is used for connection plugins (e.g. `ios` , `junos`)
- `vendor` is used strictly for logic routing
- Persistent SSH sockets disabled to avoid control path bugs

3. Port State Management

- Unified playbook `toggle_port.yml`
- Takes `interface_name` and `port_state` (`enable` / `disable`) as `-e` vars
- Branches logic based on `vendor`

4. VLAN Info Gathering

- Unified `get_vlan_info.yml`
- Pulls `show vlan` or `show vlans` , plus L3 bindings (`irb` , `SVI`)
- Saves raw output to per-host `.txt` files
- `assign_vlan.yml` creates and/or assigns L2 vlans to interfaces.

5. CoA (Change of Authorization)

- CoA handled locally via `radclient`
- Accepts parameters: `coa_host` , `coa_type` (`disconnect` / `coa`), `coa_target_type` (`mac` / `user`)
- Uses Jinja2 templates to construct CoA packets

Usage Examples

Toggle Port:

```
ansible-playbook playbooks/port/toggle_port.yml -e "interface_name=GigabitEthernet0/3 port_state=disable"
```

Get VLANs:

```
ansible-playbook playbooks/vlan/get_vlan_info.yml
```

CoA Disconnect:

```
ansible-playbook playbooks/coa/send_coa.yml -e "coa_host=10.34.10.16 coa_type=disconnect coa_target_type=mac coa_value=00:e0:4c:36:2d:53"
```

Example `hosts.yml` (Location-Based Groups)

```
all:
  children:
    istanbul:
      hosts:
        cisco-switch:
          ansible_host: 10.34.10.16
        ruijie-switch:
          ansible_host: 10.34.10.15

    mugla:
      hosts:
        juniper-switch:
          ansible_host: 10.34.10.19
```

Example `ansible.cfg`

```
[defaults]
inventory = ./hosts.yml
host_key_checking = False
transport = ssh
pipelining = False
timeout = 30
retry_files_enabled = False

[ssh_connection]
control_path = /tmp/ansible-%%h-%%p-%%r
control_path_dir = /tmp
ssh_args = -o ControlMaster=auto -o ControlPersist=60s
```

Example `group_vars/`

`cisco.yml`

```
ansible_user: admin
ansible_password: Deneme12
ansible_network_os: ios
ansible_connection: network_cli
ansible_ssh_common_args: '-o StrictHostKeyChecking=no'
ansible_become: yes
ansible_become_method: enable
```

```
ansible_become_password: Deneme12
vendor: cisco
```

`juniper.yml`

```
ansible_user: admin
ansible_ssh_pass: Deneme12
ansible_network_os: junos
ansible_connection: netconf
vendor: juniper
```

`ruijie.yml`

```
ansible_user: admin
ansible_password: Deneme12
ansible_network_os: ios
ansible_connection: network_cli
ansible_ssh_common_args: '-o StrictHostKeyChecking=no'
ansible_become: yes
ansible_become_method: enable
ansible_become_password: Deneme12
vendor: ruijie
```

Example `playbooks/interfaces/get_interfaces.yml`

```
- name: Gather interface info from all switches
  hosts: all
  gather_facts: no
  tasks:
    - name: Include vendor-specific interface task
      include_tasks: "tasks/{{ ansible_network_os }}_show_interfaces.yml"
```

Example `playbooks/interfaces/tasks/`

`ios_show_interfaces.yml`

```
- name: Show interface(s) on IOS
  ios_command:
    commands:
      - "{{ 'show interfaces ' + interface_name if interface_name is defined else 'show interfaces' }}"
  register: ios_interfaces

- name: Print IOS interface info
  debug:
    var: ios_interfaces.stdout_lines
```

`junos_show_interfaces.yml`

```
- name: Show interface(s) on JUNOS
  junipernetworks.junos.junos_command:
    commands:
      - "{{ 'show interfaces terse ' + interface_name if interface_name is defined else 'show interfaces terse' }}"
  register: junos_interfaces

- name: Print JUNOS interface info
  debug:
    var: junos_interfaces.stdout_lines
```

Example `playbooks/ports/toggle_port.yml`

```
- name: Toggle port state based on vendor
  hosts: all
  gather_facts: no
  vars:
    # Do NOT set default here — force user to supply this
    # interface_name: "" ← nope
    port_state: enable
  pre_tasks:
    - name: Fail if interface_name is not provided
      fail:
        msg: "You must specify 'interface_name', e.g. -e interface_name=GigabitEthernet1/0/5"
      when: interface_name is not defined

  tasks:
    - name: Include vendor-specific task
      include_tasks: "tasks/{{ port_state }}/{{ vendor }}_{{ port_state }}.yml"
```

Example `playbooks/ports/tasks/enable`

`ios_enable.yml`

```
- name: Enable interface on IOS
  ios_config:
    lines:
      - no shutdown
    parents: "interface {{ interface_name }}"
```

`junos_enable.yml`

```
- name: Enable interface on Junos
  junipernetworks.junos.junos_config:
    lines:
      - "delete interfaces {{ interface_name }} disable"
    comment: "Enabled by Ansible"
```

Example `playbooks/ports/tasks/disable`

`ios_disable.yml`

```
- name: Disable interface on IOS
  ios_config:
    lines:
      - shutdown
    parents: "interface {{ interface_name }}"
```

`junos_disable.yml`

```
- name: Disable interface on Junos
  junipernetworks.junos.junos_config:
    lines:
      - "set interfaces {{ interface_name }} disable"
    comment: "Disabled by Ansible"
```

Example `playbooks/vlan`

`get_vlan_info.yml`

```yaml
- name: Get VLAN info from all switches
  hosts: all
  gather_facts: no
  tasks:
    - name: Include vendor-specific VLAN task
      include_tasks: "tasks/get_vlan/{{ vendor }}_get_vlan.yml"
```

`assign_vlan.yml`

```yaml
- name: Assign VLAN on access ports
  hosts: all
  gather_facts: no
  vars:
    # Must be passed via -e "interface_name=... vlan_id=..."
  pre_tasks:
    - name: Fail if interface_name or vlan_id not provided
      fail:
        msg: "You must specify 'interface_name' and 'vlan_id', e.g. -e interface_name=GigabitEthernet1/0/17 vlan_id=90"
      when: interface_name is not defined or vlan_id is not defined

  tasks:
    - name: Include vendor-specific VLAN task
      include_tasks: "tasks/assing_vlan/{{ ansible_network_os }}_assign_vlan.yml"
```

Example `playbooks/vlan/tasks/get_vlan/`

`cisco_get_vlan.yml`

```yaml
- name: Run VLAN + interface commands on IOS
  ios_command:
    commands:
      - show vlan brief
      - show ip interface brief
  register: vlan_output

- name: Save VLAN info to local file
  delegate_to: localhost
  run_once: true
  copy:
    content: "{{ vlan_output.stdout | join('\n\n') }}"
    dest: "{{ inventory_hostname }}_vlan_facts.txt"
```

`juniper_get_vlan.yml`

```yaml
- name: Run VLAN + IRB interface commands on Junos
  junipernetworks.junos.junos_command:
    commands:
      - show vlans
      - show interfaces terse | match irb
  register: vlan_output

- name: Save Juniper VLAN info to local file
  delegate_to: localhost
  run_once: true
  copy:
    content: "{{ vlan_output.stdout | join('\n\n') }}"
    dest: "{{ inventory_hostname }}_vlan_facts.txt"
```

`ruijie_get_vlan.yml`

```
- name: Run VLAN + interface commands on Ruijie
  ios_command:
    commands:
      - show vlan
      - show ip interface brief
  register: vlan_output

- name: Save VLAN info to local file
  delegate_to: localhost
  copy:
    content: "{{ vlan_output.stdout | join('\n\n') }}"
    dest: "{{ inventory_hostname }}_vlan_facts.txt"
```

Example `playbooks/vlan/tasks/assign_vlan/`

`ios_assign_vlan.yml`

```
- name: Ensure VLAN exists (IOS)
  ios_config:
    lines:
      - vlan {{ vlan_id }}

- name: Assign VLAN to interface (IOS)
  ios_config:
    lines:
      - switchport mode access
      - switchport access vlan {{ vlan_id }}
    parents: "interface {{ interface_name }}"
```

`junos_assign_vlan.yml`

```
- name: Ensure VLAN exists (JUNOS)
  junipernetworks.junos.junos_config:
    lines:
      - set vlans VLAN{{ vlan_id }} vlan-id {{ vlan_id }}
    comment: "Ensure VLAN exists"

- name: Remove existing VLAN binding from interface
  junipernetworks.junos.junos_config:
    lines:
      - delete interfaces {{ interface_name }} unit 0 family ethernet-switching vlan
    comment: "Clear old VLAN config"

- name: Set interface to access mode with VLAN
  junipernetworks.junos.junos_config:
    lines:
      - set interfaces {{ interface_name }} unit 0 family ethernet-switching interface-mode access
      - set interfaces {{ interface_name }} unit 0 family ethernet-switching vlan members VLAN{{ vlan_id }}
    comment: "Assign VLAN to access port"
```

Example `playbooks/coa/send_coa.yml`

```yaml
- name: Send RADIUS CoA request
  hosts: localhost
  gather_facts: no
  vars:
    coa_type: "disconnect"  # or "coa"
    coa_target_type: "mac"  # or "user"
    coa_value: ""           # like "00:e0:4c:36:2d:53" or "ege"
    coa_host: ""            # target switch IP
    coa_secret: "Deneme12"

  pre_tasks:
    - name: Validate required vars
      fail:
        msg: "You must set coa_value and coa_host (MAC/User + switch IP)"
      when: coa_value == "" or coa_host == ""

  tasks:
    - name: Choose CoA attribute template
      set_fact:
        coa_template_file: "{{ 'coa_mac.tpl' if coa_target_type == 'mac' else 'coa_user.tpl' }}"

    - name: Create CoA input file
      template:
        src: "templates/{{ coa_template_file }}"
        dest: "/tmp/coa_input_{{ inventory_hostname }}.txt"

    - name: Send CoA packet with radclient
      shell: |
        radclient -x {{ coa_host }}:3799 {{ coa_type }} {{ coa_secret }} < /tmp/coa_input_{{ inventory_hostname }}.txt
      register: coa_result

    - name: Show result
      debug:
        var: coa_result.stdout_lines
```

Example `playbooks/coa/templates`

`coa_mac.tpl`

```
Calling-Station-Id = {{ coa_value }}
```

`coa_user.tpl`

```
User-Name = {{ coa_value }}
```