Ansible

Samlad information om Ansible.

- Pre & post tasks
- Filöverföring över SCP
- Taggar
- Moduler
- Loopar
- Inventory
- Serial
- Jinja2

Pre & post tasks

I main.yml kan man använda pre och post tasks som sker innan jobben som är i roles. Det kan vara bra att slänga på tagen always på dessa så att de körs oavsett vilken tag man använder för resten av jobbet.

Exempel:

```
! Bra pre och post tasks för Cisco IOS-XE

  pre_tasks:

  - name: Gather facts
    cisco.ios.ios_facts:
      gather_subset: all
    register: cisco_ios_facts
    tags: always

  - name: Configure archive function
    cisco.ios.ios_config:
      lines:
      - logging enable
      - logging size 200
      - notify syslog contenttype plaintext
      parents: [archive, log config]
    tags: always

  - name: Set path for archive
    cisco.ios.ios_config:
      lines:
      - "path flash:"
      parents: [archive]
    tags: always

  - name: Revert configuration in 30 minutes if script failure
    cisco.ios.ios_command:
```

```
    commands: configure terminal revert timer 30
  tags: always

post_tasks:
  - name: Configure confirm!
    cisco.ios.ios_command:
      commands: configure confirm
    tags: always
  - name: Save running-configuration to startup-configuration when modified
    cisco.ios.ios_config:
      save_when: modified
    tags: always
  - name: Pause and continue
    pause:
      prompt: "Complete, press enter to continue with , Ctrl+C to exit"
    tags: always
```

Här har vi ett annat bra exempel på hur man kan skriva sin main.yml med felhantering, som backar Cisco-konfen vid fel:

```
---
- name: Configure labb-switch
  hosts: labb
  gather_facts: false

  pre_tasks:

  - name: Gather facts
    cisco.ios.ios_facts:
      gather_subset: all
    register: cisco_ios_facts
    tags: always

  - name: Configure archive function
    cisco.ios.ios_config:
      lines:
      - logging enable
      - logging size 200
      - notify syslog contenttype plaintext
      parents: [archive, log config]
```

```
  tags: always

- name: Set path for archive
  cisco.ios.ios_config:
    lines:
    - "path flash:"
    parents: [archive]
  tags: always

- name: Revert configuration in 30 minutes if script failure
  cisco.ios.ios_command:
    commands: configure terminal revert timer 30
  tags: always

tasks:
  - block:
      - import_role:
          name: common
      - import_role:
          name: vlans
      - import_role:
          name: security
      - import_role:
          name: qos
      - import_role:
          name: templates
      - import_role:
          name: interfaces
      - import_role:
          name: routing
      - import_role:
          name: aaa
      - import_role:
          name: management
      - import_role:
          name: telemetry

    rescue:
      - name: Backa konfiguration vid problem
        cisco.ios.ios_command:
```

```
        commands: configure revert now
      ignore_errors: yes
      tags: always

post_tasks:
  - name: Configure confirm!
    cisco.ios.ios_command:
      commands: configure confirm
    tags: always
  - name: Save running-configuration to startup-configuration when modified
    cisco.ios.ios_config:
      save_when: modified
    tags: always
  - name: Pause and continue
    pause:
      prompt: "LABB complete, press enter to continue with next hosts, Ctrl+C to exit"
    tags: always
```

Det är då rescue som ser till att configure revert now slås vid fel i Ansible-jobbet. Rescue måste finnas i ett block, därför finns alla tasks i ett block.

Filöverföring över SCP

Jag brottades med att få filöverföring över SCP att fungera i Ansible med de inbyggda modulerna till Cisco-routrar. Det funkade inte.

Det som till slut fungerade var att slå Linux-kommandon med `ansible.builtin.command`.

```
    - name: Kopiera image från lokal maskin till routern via SCP
      ansible.builtin.command:
        cmd: sshpass -p "{{ ansible_ssh_pass }}" scp -O -o StrictHostKeyChecking=no {{ image_src_path }} {{
ansible_user }}@{{ inventory_hostname }}:{{ image_dest_path }}
      delegate_to: localhost
      when: image_file not in dir_out.stdout[0]
```

sshpass -p används här för att ta lösenordet man matar in när man kör jobbet med flaggan -k, men man kan köra utan och manuellt slå lösenord varje gång jobbet sker.

Här är ett filöverföringsjobb för att ladda över IOS-XE mjukvara och NBAR-fil:

```
---
- name: Ladda upp IOS-XE mjukvara till routrar
  hosts: all
  gather_facts: no
  vars:
    image_file: "cat9k_iosxe.17.12.06.SPA.bin"
    image_src_path: "/data/dir/c9300-c9500/17.12.6/{{ image_file }}"
    image_dest_path: "bootflash:{{ image_file }}"
    ansible_command_timeout: 7200  # 2 timmar
    nbar_file: "pp-adv-cat9k-1712.1-49-74.0.0.pack"
    nbar_src_path: "/data/dir/c9300-c9500/17.12.6/{{ nbar_file }}"
    nbar_dest_path: "bootflash:{{ nbar_file }}"

  tasks:

    - name: Print paths
      ansible.builtin.debug:
        msg:
```

```
      - "image_src_path: {{ image_src_path }}"
      - "image_dest_path: {{ image_dest_path }}"

  - name: Kontrollera om IOS-XE image redan finns
    cisco.ios.ios_command:
      commands:
        - "dir bootflash: | include {{ image_file }}"
    register: dir_out

  - name: Ta bort oanvända IOS-XE filer (install remove inactive)
    ansible.netcommon.cli_command:
      command: "install remove inactive"
      prompt: "Do you want to remove the above files\\? \\[y/n\\]"
      answer: "y"
    when: image_file not in dir_out.stdout[0]

  - name: Kopiera IOS-XE image från lokal maskin till routern via SCP
    ansible.builtin.command:
      cmd: sshpass -p "{{ ansible_ssh_pass }}" scp -O -o StrictHostKeyChecking=no {{ image_src_path }} {{
ansible_user }}@{{ inventory_hostname }}:{{ image_dest_path }}
    delegate_to: localhost
    when: image_file not in dir_out.stdout[0]

  - name: Verifiera att filen finns på routern
    cisco.ios.ios_command:
      commands:
        - "dir bootflash: | include {{ image_file }}"
    register: verify_dir
    failed_when: image_file not in verify_dir.stdout[0]



  - name: Kontrollera om NBAR-filen redan finns på routern
    cisco.ios.ios_command:
      commands:
        - "dir bootflash: | include {{ nbar_file }}"
    register: dir_out_nbar

  - name: Kopiera NBAR-fil från lokal maskin till routern via SCP
    ansible.builtin.command:
      cmd: sshpass -p "{{ ansible_ssh_pass }}" scp -O -o StrictHostKeyChecking=no {{ nbar_src_path }} {{
```

```
ansible_user }}@{{ inventory_hostname }}:{{ nbar_dest_path }}
      delegate_to: localhost
      when: dir_out_nbar.stdout[0] is search(nbar_file) == False

    - name: Verifiera att NBAR-filen finns på routern
      cisco.ios.ios_command:
        commands:
          - "dir bootflash: | include {{ nbar_file }}"
      register: verify_dir_nbar
      failed_when: verify_dir_nbar.stdout[0] is search(nbar_file) == False
```

Taggar

Man kan tilldela en funktion taggar. Ex:

```
- name: Configure dot1q interfaces
  cisco.ios.ios_l2_interfaces:
    config:
      - name: "{{ item.name }}"
        mode: trunk
        trunk:
          native_vlan: "{{ native_vlan }}"
          allowed_vlans: "{{ access_vlans }}"
    state: replaced
  loop: "{{ trunk_interfaces }}"
  when: trunk_interfaces is defined
  tags:
    - vlan
    - trunk
```

När man sen kör ansible med en tagg gör man det genom: `ansible-playbook -main.yml -t vlan,svi`

Kommatecknet mellan taggarna i kommandot ovan gör att man matchar flera taggar.

Moduler

`ansible-doc -l` listar alla moduler som Ansible känner till och har tillgång till, inklusive moduler i installerade collections.

Funktion:

- Skannar Ansible-core-moduler
- Skannar alla installerade collections
- Visar modulnamn i formatet `namespace.collection.module`
- Används för att kontrollera om en modul faktiskt finns

Exempel:

```
ansible-doc -l | grep cisco.ise.network
cisco.ise.networkdevice
...
cisco.ise.networkdevice_info
...
cisco.ise.networkdevicegroup
...
cisco.ise.networkdevicegroup_info                                                    ...
```

Loopar

Loopar byggs genom att lägga till `when` i en funktion. Se exempel:

```
- name: Configure root guard on trunk interfaces in L3 switches
  cisco.ios.ios_config:
    lines:
    - spanning-tree guard root
    parents: ["interface {{ item.name }}"]
  loop: "{{ trunk_interfaces }}"
  when: "trunk_interfaces is defined and 'dist' in group_names"
```

Då går loopen genom listan trunk_interfaces.

Konfigurerar man enligt ovan, med parents, så se till att inga extra mellanrum har smugit in. Då kommer loopen att slå kommandona 9 gånger istället för 1 gång, vilket kan göra ansible-jobbet otroligt långsamt.

Inventory

Inventory kan exempelvis specifiseras som .ini-filer. Här är ett exempel på en sådan fil:

```
[docker_mgmt_2]
docker-mgmt-2.jehrlander.net

[ubuntu:children]
docker_mgmt_2
```

Notera children för att hänvisa till andra objekt i filen för att gruppera.

Serial

Serial används för att utföra ett jobb på enbart ett visst antal enheter i en grupp åt gången.

Serial definieras .yml-filen som kör jobbet, ex:

```
---
- name: Ladda upp IOS-XE mjukvara till routrar
  hosts: routers
  serial: 20
```

Har man 1000 routrar kommer då enbart 20 köras åt gången.

Det går att använda serial på flera sätt:

```
# Med nedan så körs först 1 host, sedan 5 och för resterande som finns kvar körs 10 i taget
  serial:
    - 1
    - 5
    - 10

# Med nedan körs först 1, sedan 10% av de som finns kvar och sedan 25% av de som finns kvar tills allt är klart
  serial:
    - 1          # First: 1 host (canary)
    - "10%"      # Second: 10% of remaining
    - "25%"      # Third+: 25% of remaining

# Kombinera med max_fail_Percentage för att avsluta jobbet om för många misslyckas

  serial: 5
  # Stop if more than 20% of hosts fail
  max_fail_percentage: 20

# Avsluta jobbet om 1 fel uppstår

  serial: 1
  # Stop on first failure
  any_errors_fatal: true
```

I mitt exempel så använder jag serial för att uppgradera routrar. Då är det bra med en check om rätt mjukvara redan finns så att det inte sker på nytt, jobbet avslutas med `meta` för den här hosten om den redan finns:

```
tasks:

  - name: Kontrollera nuvarande mjukvara
    cisco.ios.ios_command:
      commands:
        - "show install active | include {{ image_version }}"
    register: current_firmware

  - name: Stoppa om version redan är installerad
    meta: end_host
    when: image_version in current_firmware.stdout[0]
```

Jinja2

Jag är inte bra på Jinja2. Här är vad jag har samlat på mig hittils.

Om man konfigurerar statiska rutter med Ansible kan man använda sig av jinja2-templates för att enbart ha med next-hop om det faktiskt finns definierat i inventory. Se exempel:

```
- name: Configure static routes
  cisco.ios.ios_config:
    lines:
      - "ip route {{ item.name }} {{ item.interface }}{% if item.next_hop is defined %} {{ item.next_hop }}{% endif %}"
  when: static_routes is defined
  loop: "{{ static_routes }}"

! Inventory/host_vars. Första exemplet med next hop och andra utan.
static_routes:
  - name:  8.8.4.4 255.255.255.255
    interface: GigabitEthernet0/0/0
    next_hop: 1.1.1.1
  - name: 8.8.4.4 255.255.255.255
    interface: GigabitEthernet0/1/0.2
```