Network Automation with Ansible

(Cisco ios network devices)

By Ershad RAMEZANI

Table of Contents

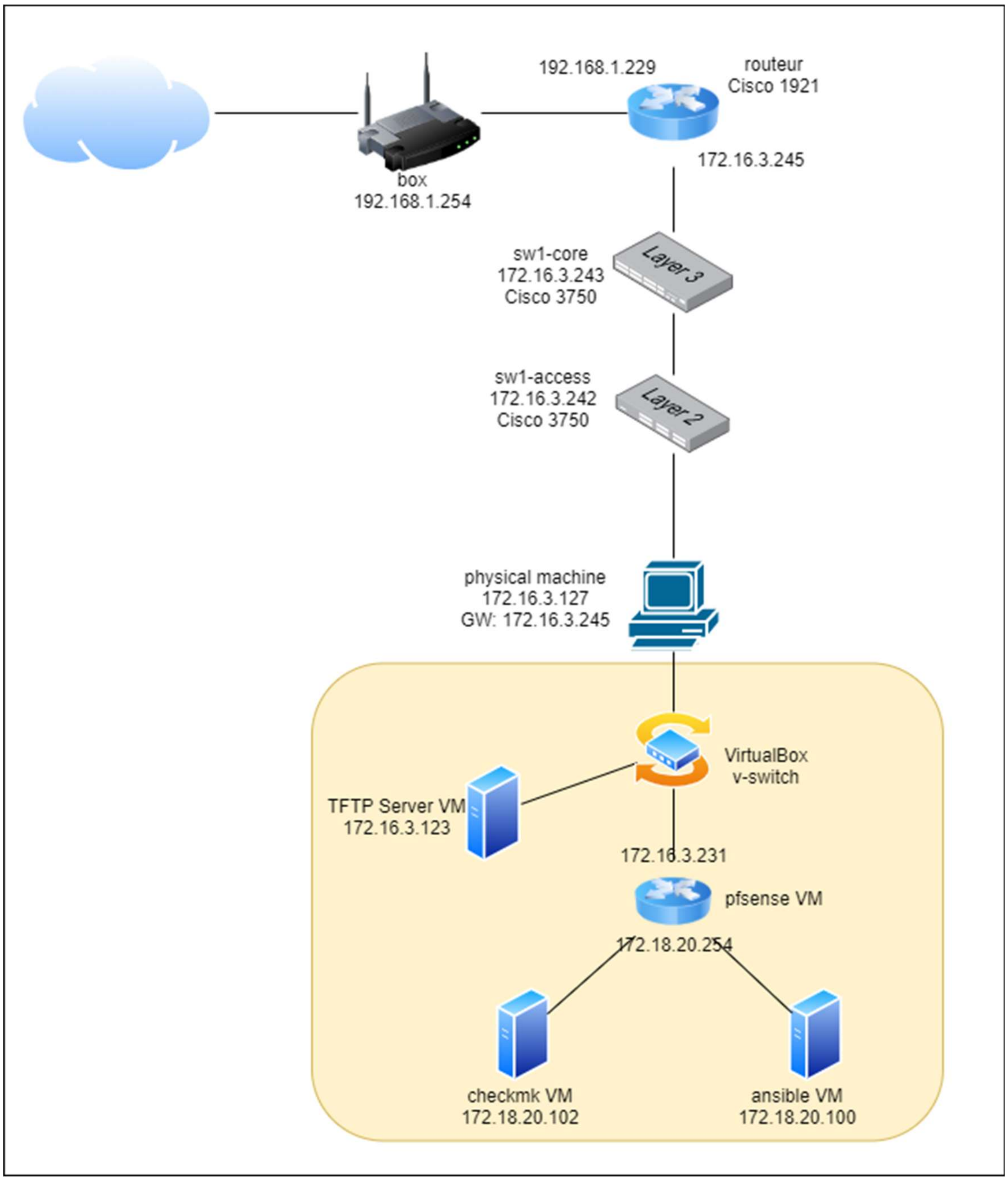
Preface

In this article we are going to learn how to automate configuration of Cisco network devices using Ansible. The main goal of this article is to give you a general overview of how to create your Ansible inventory, introduce you to the different modules that are used for network automation by giving some practical examples, discovering the *collections* that are used for network automation and different *connections* plugins required by these collections.

But first thing first, we will start by explaining the network topology that we have implemented in this article.

Topology

This topology consists of three Cisco devices. Two switches are connecting in line to a router and all three are configured in 172.16.3.0/24 network. We also have three VMs on a VirtualBox hypervisor with host-only network configuration in 172.18.20.0/24 network. We will use ansible server VM to configure the network devices such as backing up running configuration and the image files into TFTP server and configuring SNMP protocol on network devices in order to monitor them using *CheckMK* monitoring server.



Ansible installation

I'm going to install the ansible core version 2.3 on an ubuntu machine. Ansible installation is as simple as these three commands (plus apt update and apt upgrade). Please refer to this link for Ansible documentation:

https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html

```
sudo apt update & sudo apt upgrade
sudo apt install software-properties-common
sudo apt-add-repository --yes --update ppa:ansible/ansible
sudo apt update
sudo apt install ansible -y
```

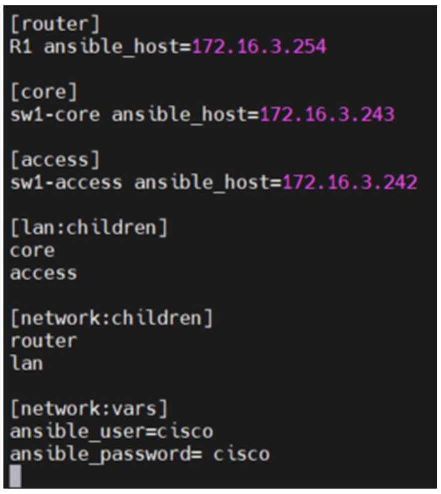
The ansible root directory will be created in */etc/ansible* which by default contains the *ansible.cfg* file, the *hosts* file and the empty *roles* folder. Ansible.cfg is the configuration file where ansible will look for its configuration parameters. Hosts file is the default inventory for all the hosts that can be managed by Ansible.

We can create our project in the directory of our choice. I am going to create it in my home directory.

```
cd /home/ansible
mkdir cisco_ios_project
```

inventory

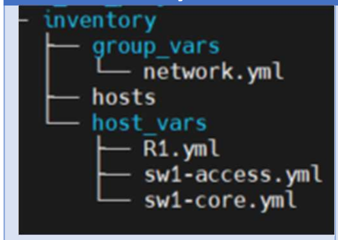
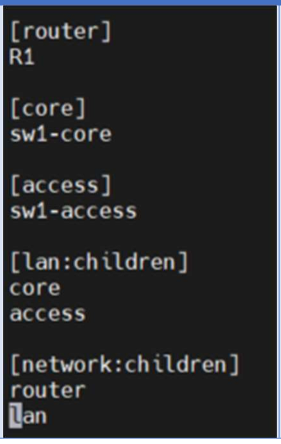
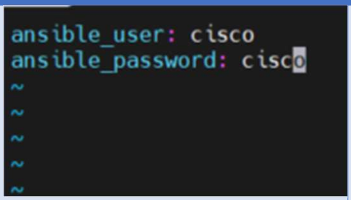
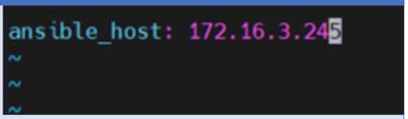
The inventory hosts file is in .ini format and contains all the hosts that can be managed by ansible. Hosts can be written as a single host name (aliases) or as a group. In addition to theirs names, we can add different variables for each or a group of hosts. For example adding *ansible_host* variable for each host to define the FQND or IP address of the host to guaranty its reachability within the ansible infrastructure. Or the *ansible_user* variable for a group to define the user with whom Ansible can connect to a group of hosts.



```
[router]
R1 ansible_host=172.16.3.254

[core]
sw1-core ansible_host=172.16.3.243

[access]
sw1-access ansible_host=172.16.3.242

[lan:children]
core
access

[network:children]
router
lan

[network:vars]
ansible_user=cisco
ansible_password= cisco
```

Note: All codes in this article are available in my github at https://github.com/ershad-ra/ansible_ios_automation

But in a large network infrastructure, this kind of inventory files can get soon complicated to manage. In order to facilitate the inventory management, we can use another method based on inventory directories and files. In this method, beside the hosts file which contains hosts and groups, we will create two folders named *group_vars* and *host_vars*. The group_vars contains the files for group variables and host_vars contain the hosts specific variables. These files are in *yaml* format (unlike the hosts file) and are named according to their corresponding host or group name in the hosts file.

Tree command on inventory folder	Hosts file	Group_vars/Network.yml	Host_vars/R1.yml
			

```
├ inventory
│ ├── group_vars
│ │   └── network.yml
│ ├── hosts
│ └── host_vars
│     ├── R1.yml
│     ├── sw1-access.yml
│     └── sw1-core.yml
```

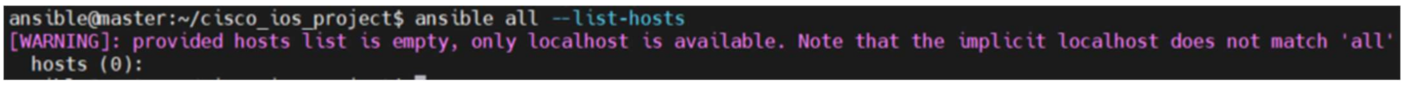
```
[router]
R1

[core]
sw1-core

[access]
sw1-access

[lan:children]
core
access

[network:children]
router
lan
```

```
ansible_user: cisco
ansible_password: cisco
~
~
~
~
~
```

```
ansible_host: 172.16.3.245
~
~
~
```

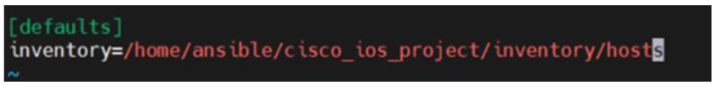
Ansible.cfg file

If we execute the ansible all --list-hosts command in the CLI, we will get the following message:

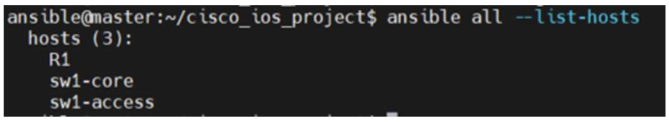


```
ansible@master:~/cisco_ios_project$ ansible all --list-hosts
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'
  hosts (0):
```

This warning message means that there is no hosts configured in the default hosts file. And only localhost is available (Ansible by default adds localhost to this list). That is because the default location for inventory hosts file is */etc/ansible/hosts*. To change this default behavior, we will create a new ansible.cfg file in our project directory and reconfigure this default parameter.

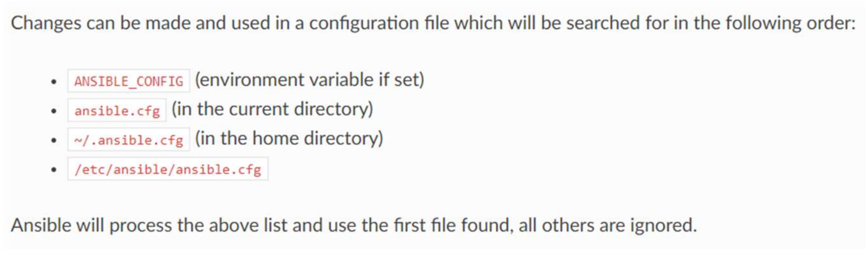


```
[defaults]
inventory=/home/ansible/cisco_ios_project/inventory/hosts
~
```



```
ansible@master:~/cisco_ios_project$ ansible all --list-hosts
  hosts (3):
    R1
    sw1-core
    sw1-access
```

Note that according to Ansible precedence rules, the ansible.cfg file located in the current directory where we are executing the ansible command has the highest priority, and the home directory sits in second place. So although there is an ansible.cfg file located in */etc/ansible* directory, our project specific ansible.cfg file will be considered.



Changes can be made and used in a configuration file which will be searched for in the following order:

- `ANSIBLE_CONFIG` (environment variable if set)
- `ansible.cfg` (in the current directory)
- `~/.ansible.cfg` (in the home directory)
- `/etc/ansible/ansible.cfg`

Ansible will process the above list and use the first file found, all others are ignored.

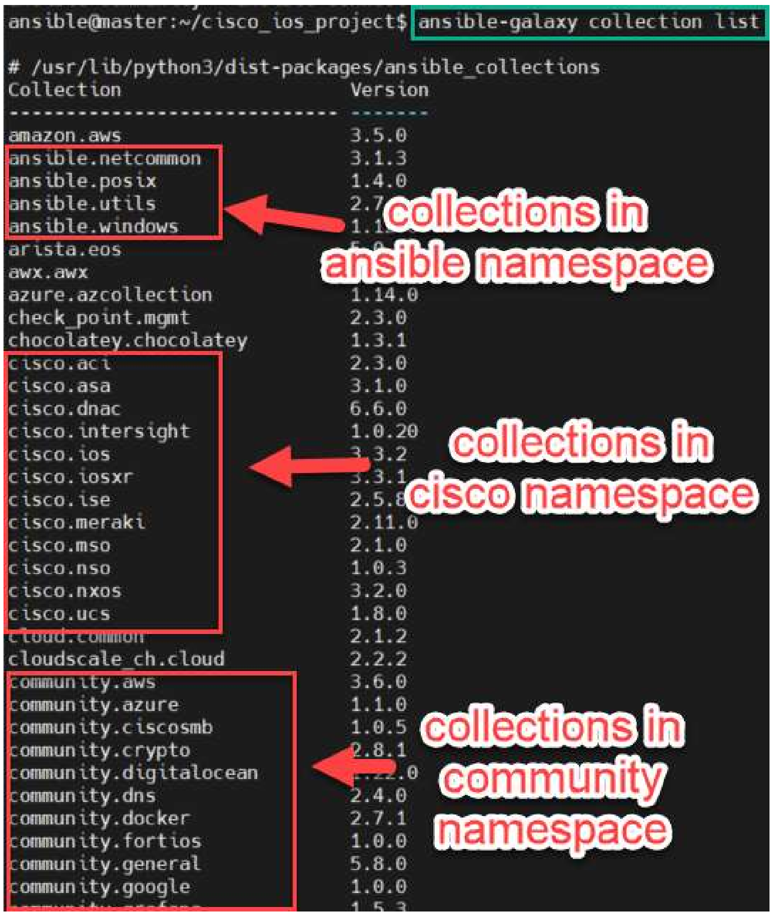
Modules, plugins and collections

Ansible modules are units of code that can control system resources or execute system commands. Modules can be executed directly on remote hosts with ad-hoc commands or through playbooks. almost all the ansible modules are written in python languages (some of Windows modules are written in PowerShell). That is why python installation and its version on ansible controller and the nodes is important.

Similar to modules are plugins, which are pieces of code that extend core Ansible functionality.

A list of modules and plugins can come together and make a collection. Collections are a distribution format for Ansible content that can include playbooks, roles, modules and plugins. We can install collections through a distribution server such as Ansible Galaxy. The command for installing collections is *ansible-galaxy*.

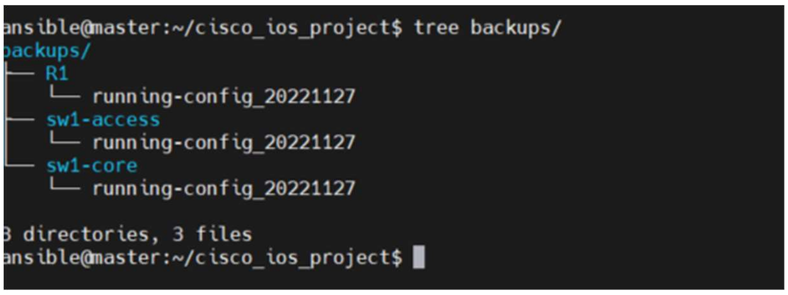
Ansible core already comes with some preinstalled collections. We can list these collections with *ansible-galaxy collection list* command. This picture is just a snippet of the collections prompt in my CLI:



```
ansible@master:~/cisco_ios_project$ ansible-galaxy collection list

# /usr/lib/python3/dist-packages/ansible_collections
Collection                    Version
----------------------------- -------
amazon.aws                    3.5.0
ansible.netcommon             3.1.3
ansible.posix                 1.4.0
ansible.utils                 2.7
ansible.windows               1.1
arista.eos                    5.0
awx.awx
azure.azcollection            1.14.0
check_point.mgmt              2.3.0
chocolatey.chocolatey         1.3.1
cisco.aci                     2.3.0
cisco.asa                     3.1.0
cisco.dnac                    6.6.0
cisco.intersight              1.0.20
cisco.ios                     3.3.2
cisco.iosxr                   3.3.1
cisco.ise                     2.5.8
cisco.meraki                  2.11.0
cisco.mso                     2.1.0
cisco.nso                     1.0.3
cisco.nxos                    3.2.0
cisco.ucs                     1.8.0
cloud.common                  2.1.2
cloudscale_ch.cloud           2.2.2
community.aws                 3.6.0
community.azure               1.1.0
community.ciscosmb            1.0.5
community.crypto              2.8.1
community.digitalocean        1.22.0
community.dns                 2.4.0
community.docker              2.7.1
community.fortios             1.0.0
community.general             5.8.0
community.google              1.0.0
```

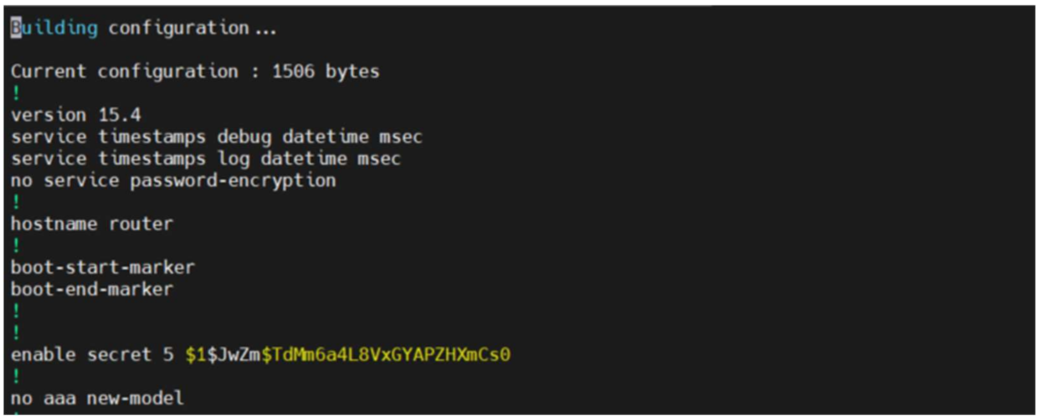
collections in ansible namespace
collections in cisco namespace
collections in community namespace

Modules are gathered into collections such as *netcommon*, *windows*, *ios*, etc. and collections are gathered into namespaces such as *cisco*,



```
ansible@master:~/cisco_ios_project$ tree backups/
backups/
├── R1
│   └── running-config_20221127
├── sw1-access
│   └── running-config_20221127
└── sw1-core
    └── running-config_20221127

3 directories, 3 files
ansible@master:~/cisco_ios_project$
```

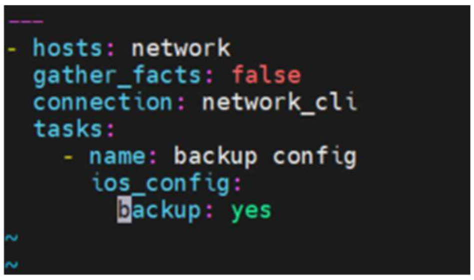
And the content of *R1/running-config_20221127*:



```
Building configuration...

Current configuration : 1506 bytes
!
version 15.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname router
!
boot-start-marker
boot-end-marker
!
!
enable secret 5 $1$JwZm$TdMm6a4L8VxGYAPZHXmCs0
!
no aaa new-model
```

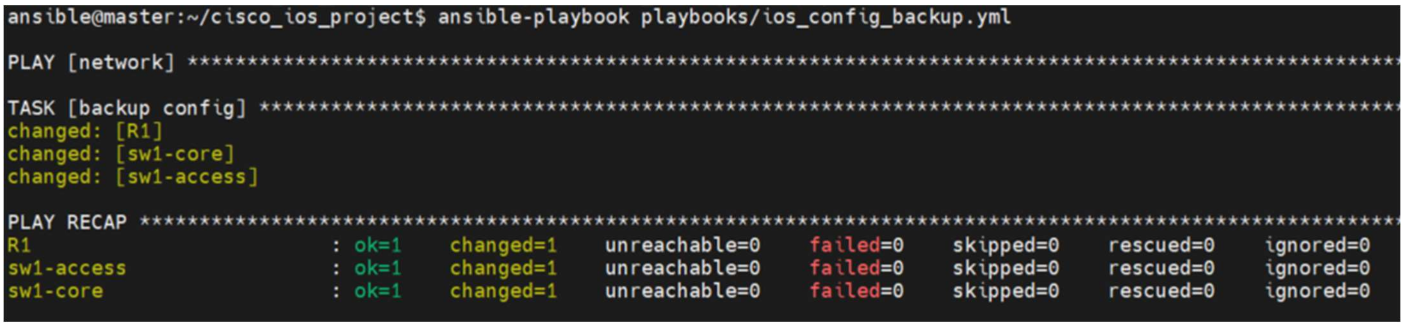
Back up running configuration with ios_config playbook

In this playbook we are going to use *ios_config* module to create a playbook for automating running configuration backup.

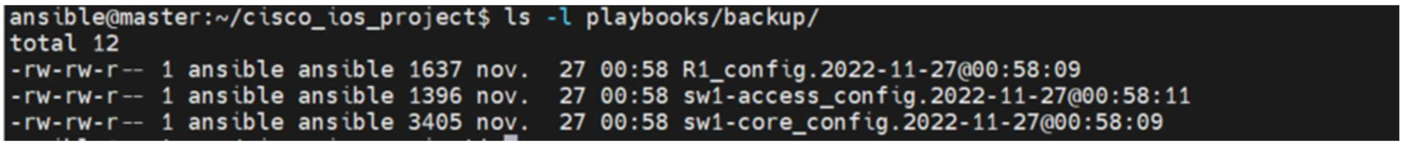


```
---
- hosts: network
  gather_facts: false
  connection: network_cli
  tasks:
    - name: backup config
      ios_config:
        backup: yes
~
~
```

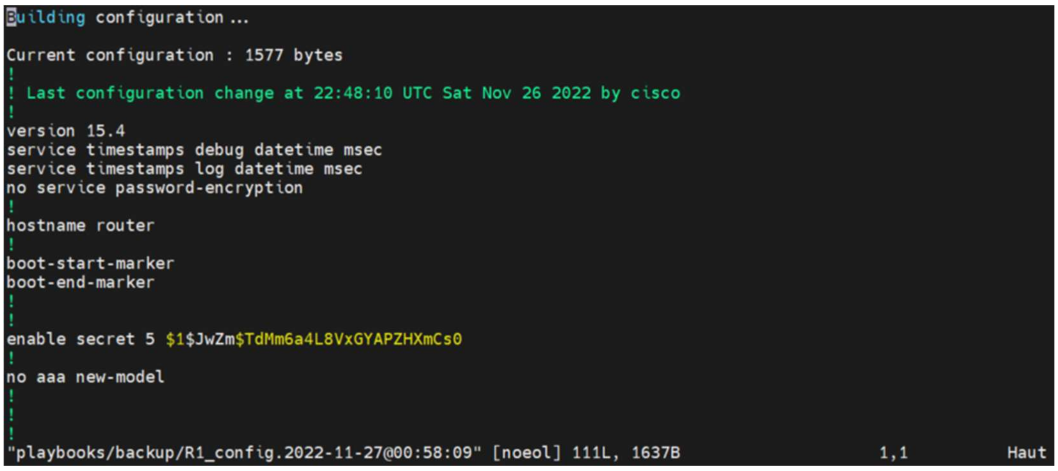
This module by default will create a backup folder in the playbook's current directory and backup the running config of each device into a separate file and name it will device hostname and the timestamp at the time of execution. Here is the result:



```
ansible@master:~/cisco_ios_project$ ansible-playbook playbooks/ios_config_backup.yml

PLAY [network] ************************************************************************************************

TASK [backup config] ******************************************************************************************
changed: [R1]
changed: [sw1-core]
changed: [sw1-access]

PLAY RECAP ****************************************************************************************************
R1                         : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
sw1-access                 : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
sw1-core                   : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```



```
ansible@master:~/cisco_ios_project$ ls -l playbooks/backup/
total 12
-rw-rw-r-- 1 ansible ansible 1637 nov.  27 00:58 R1_config.2022-11-27@00:58:09
-rw-rw-r-- 1 ansible ansible 1396 nov.  27 00:58 sw1-access_config.2022-11-27@00:58:11
-rw-rw-r-- 1 ansible ansible 3405 nov.  27 00:58 sw1-core_config.2022-11-27@00:58:09
```



```
Building configuration...

Current configuration : 1577 bytes
!
! Last configuration change at 22:48:10 UTC Sat Nov 26 2022 by cisco
!
version 15.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname router
!
boot-start-marker
boot-end-marker
!
!
enable secret 5 $1$JwZm$TdMm6a4L8VxGYAPZHXmCs0
!
no aaa new-model
!
!
!
"playbooks/backup/R1_config.2022-11-27@00:58:09" [noeol] 111L, 1637B                 1,1          Haut
```

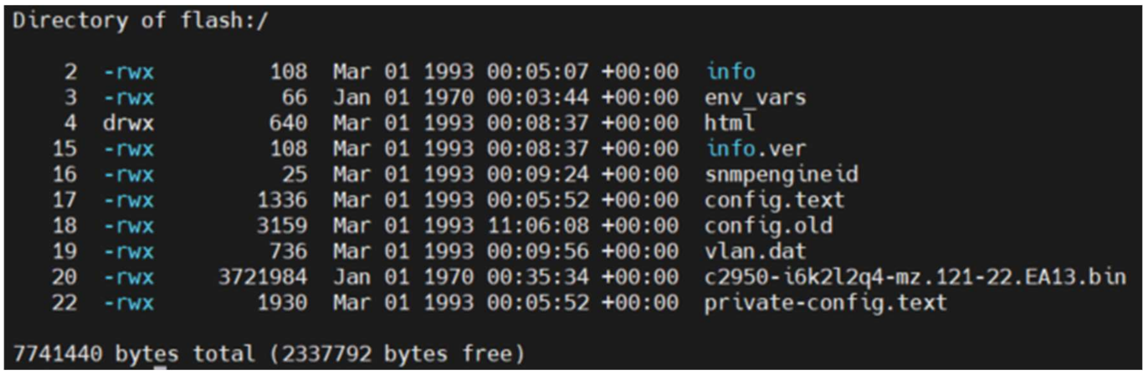
As you can see this method is much more simpler.

Backup ios images to TFTP server

If we want to upgrade the ios image of our network devices, it is always a good practice to back up the existing images before the upgrade. We can back up the ios images with the help of a TFTP server.

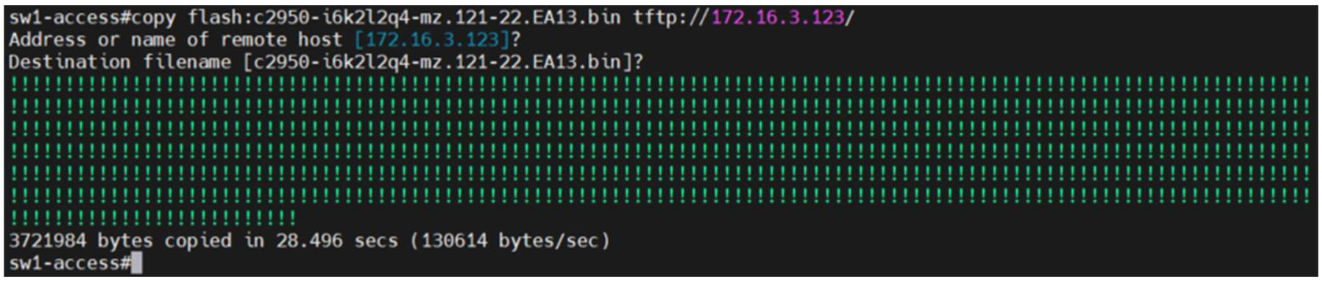
First let's see how the backup works manually through CLI on a device:

With *dir flash:* command in enable mode, we can verify the files existing in the device's flash:

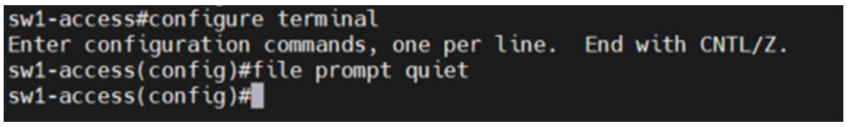


```
Directory of flash:/

    2  -rwx         108  Mar 01 1993 00:05:07 +00:00  info
    3  -rwx          66  Jan 01 1970 00:03:44 +00:00  env_vars
    4  drwx         640  Mar 01 1993 00:08:37 +00:00  html
   15  -rwx         108  Mar 01 1993 00:08:37 +00:00  info.ver
   16  -rwx          25  Mar 01 1993 00:09:24 +00:00  snmpengineid
   17  -rwx        1336  Mar 01 1993 00:05:52 +00:00  config.text
   18  -rwx        3159  Mar 01 1993 11:06:08 +00:00  config.old
   19  -rwx         736  Mar 01 1993 00:09:56 +00:00  vlan.dat
   20  -rwx     3721984  Jan 01 1970 00:35:34 +00:00  c2950-i6k2l2q4-mz.121-22.EA13.bin
   22  -rwx        1930  Mar 01 1993 00:05:52 +00:00  private-config.text

7741440 bytes total (2337792 bytes free)
```

The image has *.bin* extension. To copy it to TFTP server, type the command below and the image uploads to TFTP server. As you can see switch prompt some questions which are not suitable for automation. We can disable is with file prompt quiet command in configuration terminal mode:

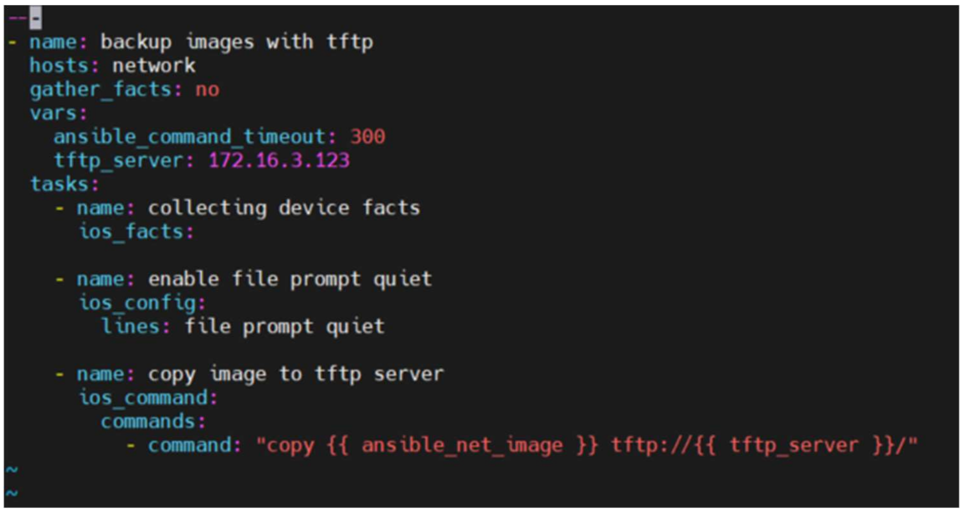


```
sw1-access#copy flash:c2950-i6k2l2q4-mz.121-22.EA13.bin tftp://172.16.3.123/
Address or name of remote host [172.16.3.123]?
Destination filename [c2950-i6k2l2q4-mz.121-22.EA13.bin]?
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!
3721984 bytes copied in 28.496 secs (130614 bytes/sec)
sw1-access#
```



```
sw1-access#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
sw1-access(config)#file prompt quiet
sw1-access(config)#
```

Now let's create a playbook to automate this action for all our devices:



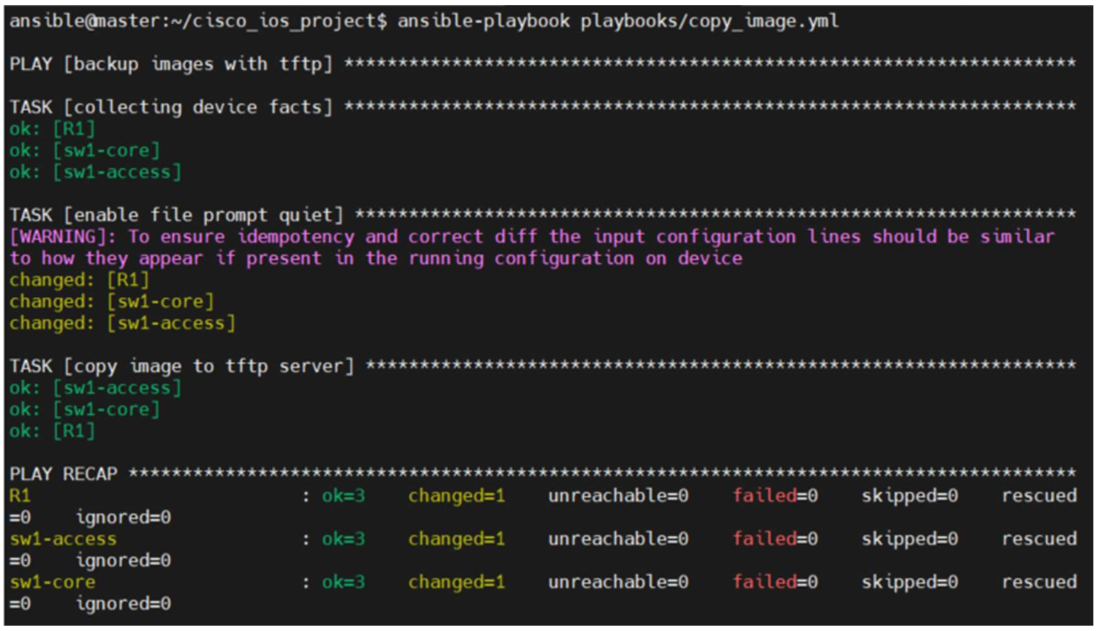
```
---
- name: backup images with tftp
  hosts: network
  gather_facts: no
  vars:
    ansible_command_timeout: 300
    tftp_server: 172.16.3.123
  tasks:
    - name: collecting device facts
      ios_facts:

    - name: enable file prompt quiet
      ios_config:
        lines: file prompt quiet

    - name: copy image to tftp server
      ios_command:
        commands:
          - command: "copy {{ ansible_net_image }} tftp://{{ tftp_server }}/"
~
~
```

This playbook has three tasks. The first one will gather facts which includes magic variables such as *ansible_net_image* (the location of the image in flash:). The second task will execute file prompt quiet. And the last one will execute the copy command on the device. The playbook perform the tasks in order and each task in parallel on all the devices in network group.

Note that the default command timeout which is the amount of time ansible should wait for response from the remote device, is set to 30 seconds. So we need to increase this amount to ensure that ansible will wait enough till all the images have been copied into TFTP server. To change this default for ansible we need to add this section in ansible file:

```
[persistent_connection]
command_timeout=200
```

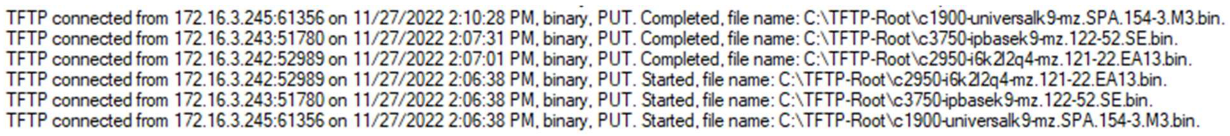
here is the result of executing this playbook:



```
ansible@master:~/cisco_ios_project$ ansible-playbook playbooks/copy_image.yml

PLAY [backup images with tftp] ********************************************************************

TASK [collecting device facts] ********************************************************************
ok: [R1]
ok: [sw1-core]
ok: [sw1-access]

TASK [enable file prompt quiet] *******************************************************************
[WARNING]: To ensure idempotency and correct diff the input configuration lines should be similar
to how they appear if present in the running configuration on device
changed: [R1]
changed: [sw1-core]
changed: [sw1-access]

TASK [copy image to tftp server] ******************************************************************
ok: [sw1-access]
ok: [sw1-core]
ok: [R1]

PLAY RECAP ****************************************************************************************
R1                         : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued
=0    ignored=0
sw1-access                 : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued
=0    ignored=0
sw1-core                   : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued
=0    ignored=0
```

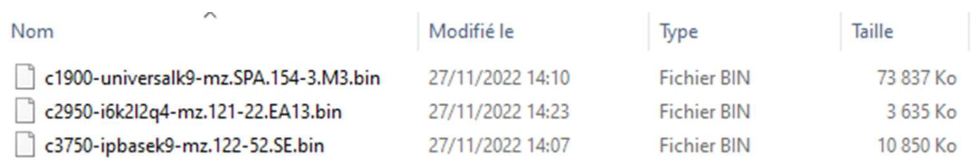
We can ignore this warning as the playbook execution was successful.

In TFTP Server log:



```
TFTP connected from 172.16.3.245:61356 on 11/27/2022 2:10:28 PM, binary, PUT. Completed, file name: C:\TFTP-Root\c1900-universalk9-mz.SPA.154-3.M3.bin.
TFTP connected from 172.16.3.243:51780 on 11/27/2022 2:07:31 PM, binary, PUT. Completed, file name: C:\TFTP-Root\c3750-ipbasek9-mz.122-52.SE.bin.
TFTP connected from 172.16.3.242:52989 on 11/27/2022 2:07:01 PM, binary, PUT. Completed, file name: C:\TFTP-Root\c2950-i6k2l2q4-mz.121-22.EA13.bin.
TFTP connected from 172.16.3.242:52989 on 11/27/2022 2:06:38 PM, binary, PUT. Started, file name: C:\TFTP-Root\c2950-i6k2l2q4-mz.121-22.EA13.bin.
TFTP connected from 172.16.3.243:51780 on 11/27/2022 2:06:38 PM, binary, PUT. Started, file name: C:\TFTP-Root\c3750-ipbasek9-mz.122-52.SE.bin.
TFTP connected from 172.16.3.245:61356 on 11/27/2022 2:06:38 PM, binary, PUT. Started, file name: C:\TFTP-Root\c1900-universalk9-mz.SPA.154-3.M3.bin.
```

Files backed up into server:



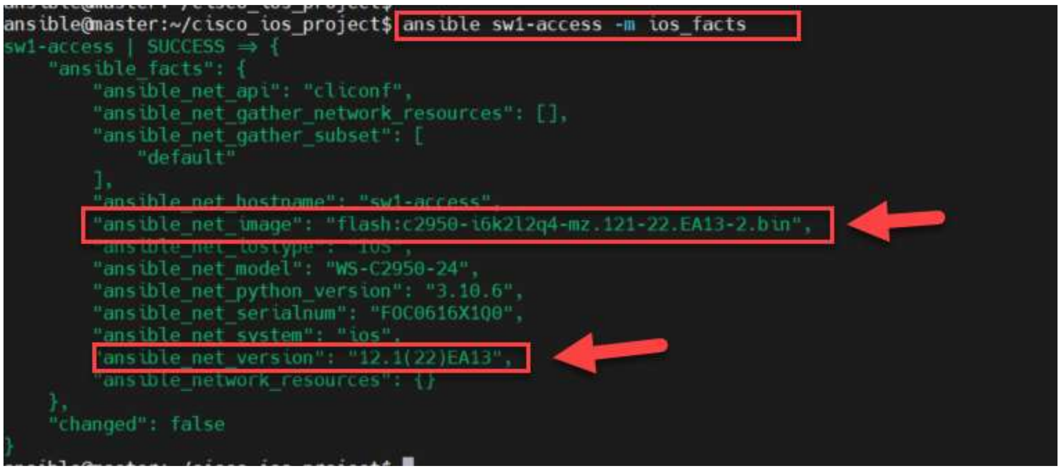
Nom	Modifié le	Type	Taille
c1900-universalk9-mz.SPA.154-3.M3.bin	27/11/2022 14:10	Fichier BIN	73 837 Ko
c2950-i6k2l2q4-mz.121-22.EA13.bin	27/11/2022 14:23	Fichier BIN	3 635 Ko
c3750-ipbasek9-mz.122-52.SE.bin	27/11/2022 14:07	Fichier BIN	10 850 Ko

Upgrading cisco ios image playbook

After backing up the current ios image, we would like to upgrade the image to a recent version. In order to do that we will create a playbook consist of three different plays.

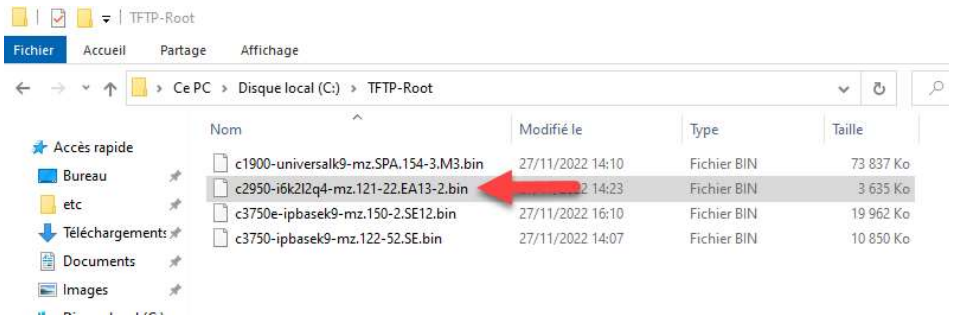
This playbook is based on the work of Roger Perkin: https://www.youtube.com/watch?v=hLhHZ_uju2Q

I am going to execute this playbook on the sw1-access. If we check the ios_fact result of this switch we can find out the ios version and the image name:



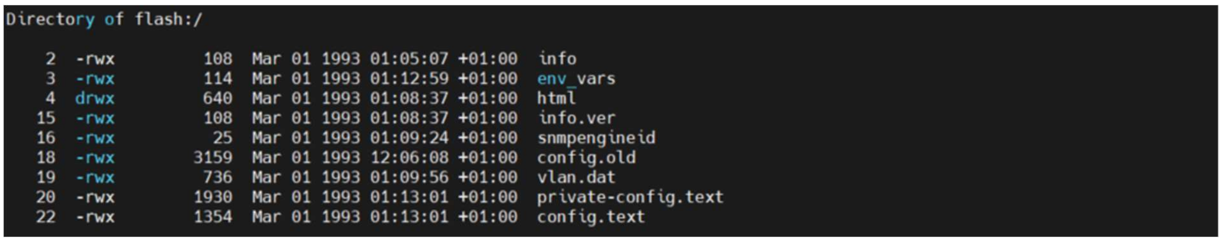
```
ansible@master:~/cisco_ios_project$ ansible sw1-access -m ios_facts
sw1-access | SUCCESS => {
    "ansible_facts": {
        "ansible_net_api": "cliconf",
        "ansible_net_gather_network_resources": [],
        "ansible_net_gather_subset": [
            "default"
        ],
        "ansible_net_hostname": "sw1-access",
        "ansible_net_image": "flash:c2950-i6k2l2q4-mz.121-22.EA13-2.bin",
        "ansible_net_iostype": "IOS",
        "ansible_net_model": "WS-C2950-24",
        "ansible_net_python_version": "3.10.6",
        "ansible_net_serialnum": "FOC0616X1Q0",
        "ansible_net_system": "ios",
        "ansible_net_version": "12.1(22)EA13",
        "ansible_network_resources": {}
    },
    "changed": false
}
```

We have already backed up this image in the previous section and I will upload the same image into TFTP server:



Nom	Modifié le	Type	Taille
c1900-universalk9-mz.SPA.154-3.M3.bin	27/11/2022 14:10	Fichier BIN	73 837 Ko
c2950-i6k2l2q4-mz.121-22.EA13-2.bin	27/11/2022 14:23	Fichier BIN	3 635 Ko
c3750e-ipbasek9-mz.150-2.SE12.bin	27/11/2022 16:10	Fichier BIN	19 962 Ko
c3750-ipbasek9-mz.122-52.SE.bin	27/11/2022 14:07	Fichier BIN	10 850 Ko

I've also deleted the current *.bin* file from the flash to make space for the new upload (that could be done in playbook as well):



```
Directory of flash:/

    2  -rwx         108  Mar 01 1993 01:05:07 +01:00  info
    3  -rwx         114  Mar 01 1993 01:12:59 +01:00  env_vars
    4  drwx         640  Mar 01 1993 01:08:37 +01:00  html
   15  -rwx         108  Mar 01 1993 01:08:37 +01:00  info.ver
   16  -rwx          25  Mar 01 1993 01:09:24 +01:00  snmpengineid
   18  -rwx        3159  Mar 01 1993 12:06:08 +01:00  config.old
   19  -rwx         736  Mar 01 1993 01:09:56 +01:00  vlan.dat
   20  -rwx        1930  Mar 01 1993 01:13:01 +01:00  private-config.text
   22  -rwx        1354  Mar 01 1993 01:13:01 +01:00  config.text
```

The first play of the playbook verifies if the *ansible_net_version* is matching the *upgrade_ios_version*:

As my switch already has the latest version of the ios image and this lab is just for a test, I am just going to add a different name than the *ansible_net_version* to simulate the need for an upgrade:

La version complète est disponible aussi mais protéger par un mot de passe.

Merci de me contacter par email :

Ershad.ra@gmail.com