
Ansible Quick Start Guide

Release 1.0

Syed Asif

Nov 17, 2022

TABLE OF CONTENTS:

1	Ansible Basic Architecture	3
1.1	Ansible Component	3
2	Ansible GNS3 Lab Setup	5
2.1	Microsoft loop-back adapter Installation	5
2.2	GNS3 and Networking device	6
2.3	Windows defender firewall setting	8
2.4	WSL-1 Installation	9
3	Ansible Installation	11
3.1	Ansible Ad-hoc Command	11
3.2	Ansible modules	12
3.3	Ansible Raw module	12
4	Ansible Inventory	15
4.1	Inventory file	15
4.2	Group of groups	16
4.3	Default groups	16
4.4	Adding Variables	16
4.5	Build Inventory for Lab	17
4.6	Ansible Inventory in YAML	17
5	Ansible Configuration	19
5.1	Gathering Facts	20
5.2	Host Key Checking	20

Warning: This is a quick start guide for Ansible and is under development.

Ansible Network modules extend the benefits of simple, powerful, agentless automation to network administrators and teams. Ansible Network modules can configure your network stack, test and validate existing network state, and discover and correct network configuration drift.

Warning: This is a quick start guide for Ansible and is under development.

ANSIBLE BASIC ARCHITECTURE

Ansible started by managing servers before expanding its ability to manage networking equipment. Ansible modules support a wide range of vendors, device types, and actions, so you can manage your entire network with a single automation tool.

With Ansible you can:

- Work without installing an agent on managed hosts.
- Automate repetitive tasks to speed up routine network changes.
- Perform changes using Python modules that run on the control.
- Communicate securely with network hardware over SSH or HTTPS.
- Benefit from the community and vendor-generated sample playbooks and roles.

1.1 Ansible Component

This Section introduces basic Ansible concepts and guides you through your first Ansible commands, playbooks and inventory entries. These concepts are common to all uses of Ansible, including network automation. You need to understand them to use Ansible for network automation.

- Control node. The Ansible server from which the control tasks take place. You can use any computer that has Python installed on it as a control node. However, you cannot use a Windows machine as a control node but on WSL (Windows Sub System for Linux) can install Ansible.
- Manage node. The network devices (and/or servers) you manage with Ansible.
- Inventory - inventory file. This file describes hosts, groups of hosts, and variables. Your inventory can specify information like IP address for each managed node.
- Playbook - script file. Ordered lists of tasks can include variables written in YAML and are easy to read, write, share and understand.
- Play - script (set of tasks). Associates tasks with hosts for which these tasks
- Task - task. Calls the module with the specified parameters and variables
- Module. Implements certain functions. Each module has a particular use, from administering users on a specific type of database to managing VLAN interfaces on a specific type of network device.

ANSIBLE GNS3 LAB SETUP

In this section, we will set up, lab topology with a Windows machine (on WSL) and GNS3. We are using a Microsoft loop-back adapter for this lab, which simplifies the management of network devices for network automation with GNS3.

The required steps for this lab are described below:

- GNS3 and VMware
- Download the PuTTY.exe file.
- Install Microsoft loop-back adapter and GNS3 configuration.
- Configuration of the network device.
- Configuration of windows defender firewall.
- WSL-1 (Windows Sub System for Linux).

2.1 Microsoft loop-back adapter Installation

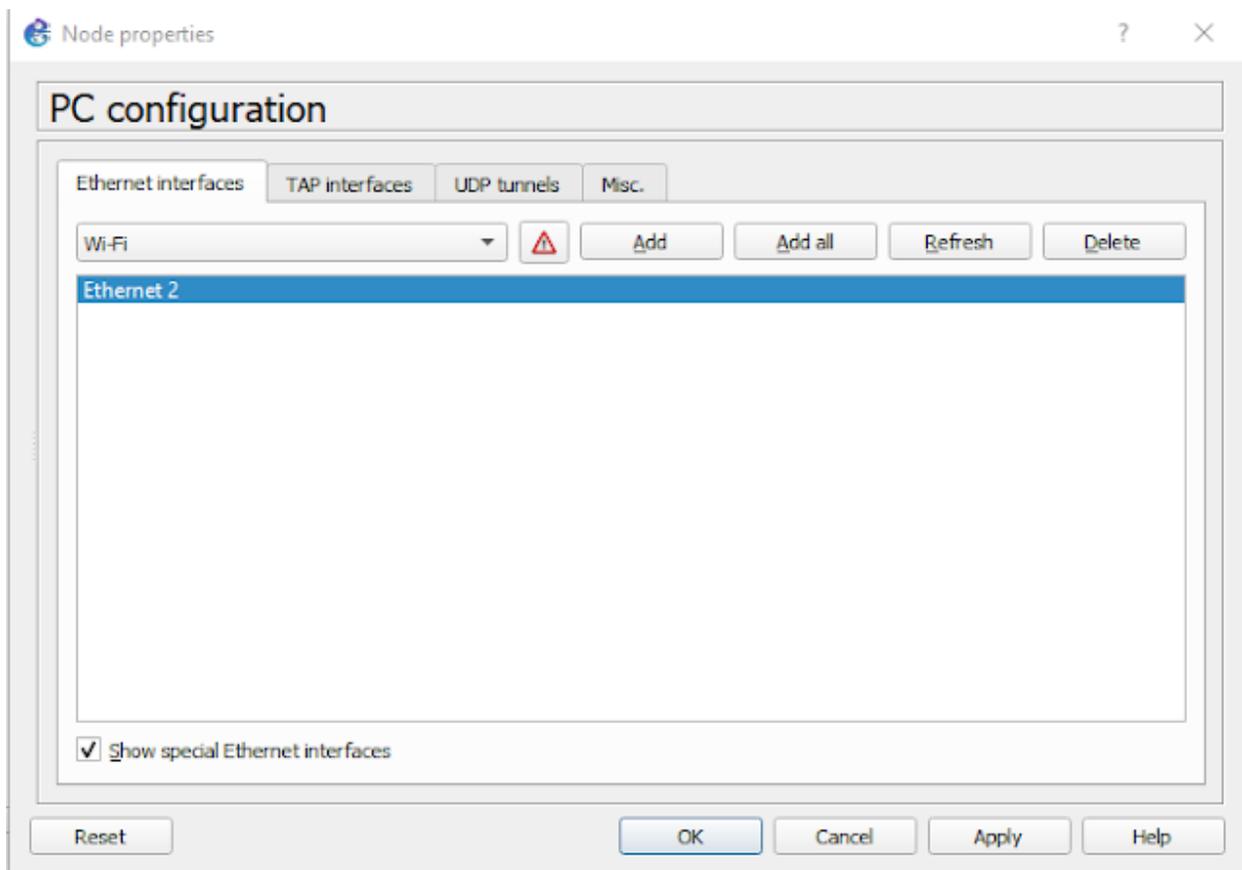
1. Right-click on the window start menu icon and select Device Manager. The device manager window will open.
2. Click on Action, and select Add legacy hardware.
3. Click next on the welcome screen.
4. Choose “Install the hardware that I manually select from a list” and click on Next.
5. Scroll down and select Network adapters from offered common hardware types and click on Next.
6. Select Microsoft as the manufacturer, and then select Microsoft KM-TEST Loop-back adapter card model, click on Next.
7. Click on Next.
8. Click on Finish and identify your loop-back adapter, in my case Ethernet2 as shown below:



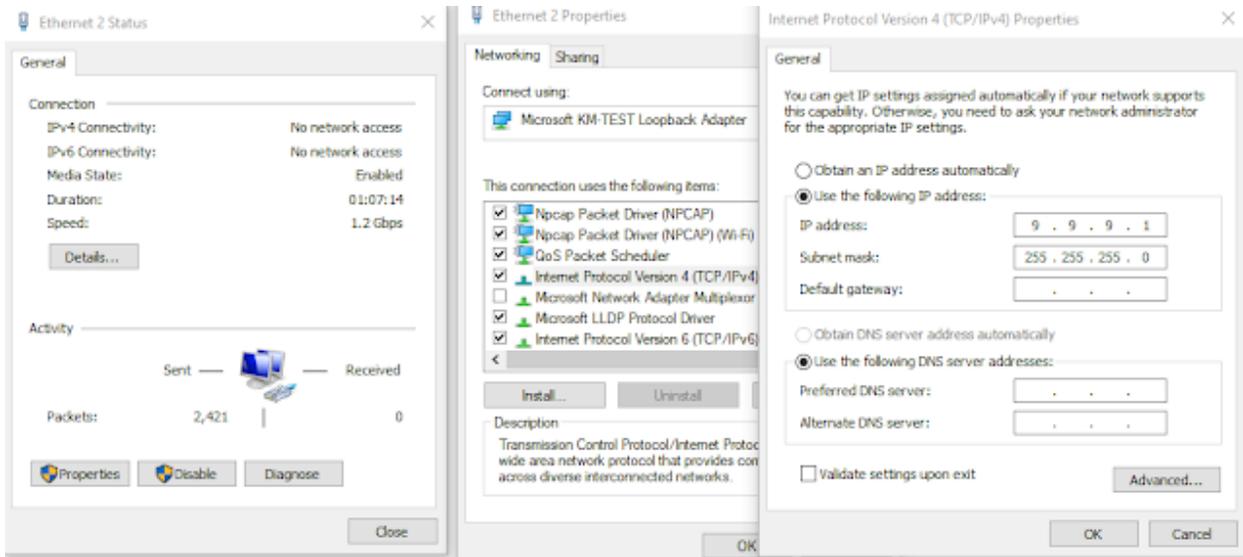
2.2 GNS3 and Networking device

Start GNS3 and drop a cloud node at the workspace and configure as below:

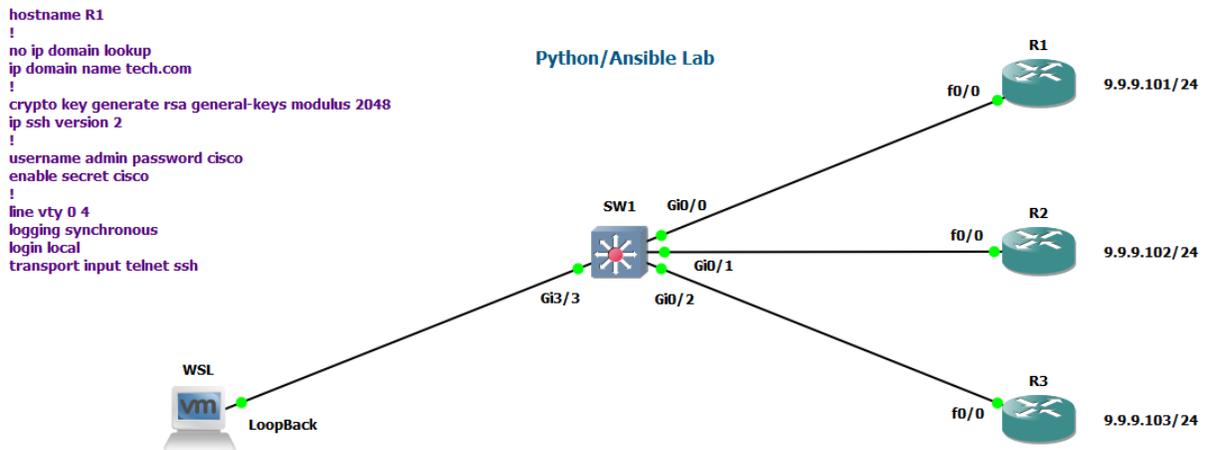
1. Right-click Cloud node, select Configure, and then select “Show special Ethernet interfaces”.
2. Add the Ethernet2 (loopback adapter) from the drop-down menu.
3. Delete any other interfaces.
4. Highlight the Ethernet2
5. Click on add button.
6. Apply OK.
7. Change the symbol to a computer and rename the cloud node to PC.



Configure Ethernet2 adapter from opening windows Network and Sharing Center then change adapter setting and add static IP address as below. The IP address used in this example is only for the lab.



Add switch and router as per below topology:



Configure SSH and device IP on SW1 as below:

```

conf t
!
hostname SW1
!
enable secret cisco
service password-encryption
!
username admin password cisco
!
no ip domain-lookup
!
ip domain name tech.com
!
crypto key generate rsa general-keys modulus 2048
!

```

(continues on next page)

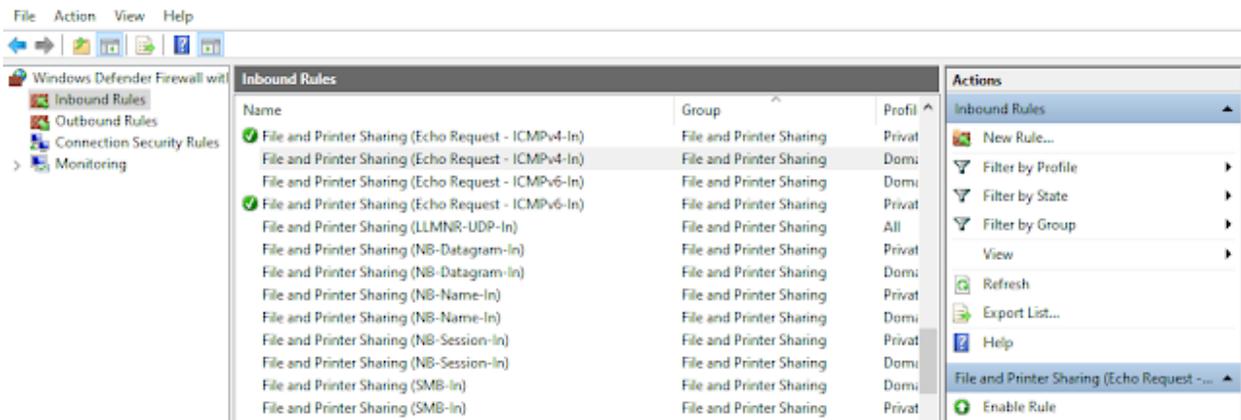
(continued from previous page)

```
interface Vlan1
 ip address 9.9.9.10 255.255.255.0
 no shutdown
!
line vty 0 4
 logging synchronous
 login local
 transport input all
!
end
!
```

Also configure routers as per the above network topology.

2.3 Windows defender firewall setting

- Open cmd.exe and check connectivity with ping, hope all is well.
- You can open the Windows Defender firewall as shown.
- Open Windows defender firewall with Advanced Security and enable these two settings on the inbound rules as below.



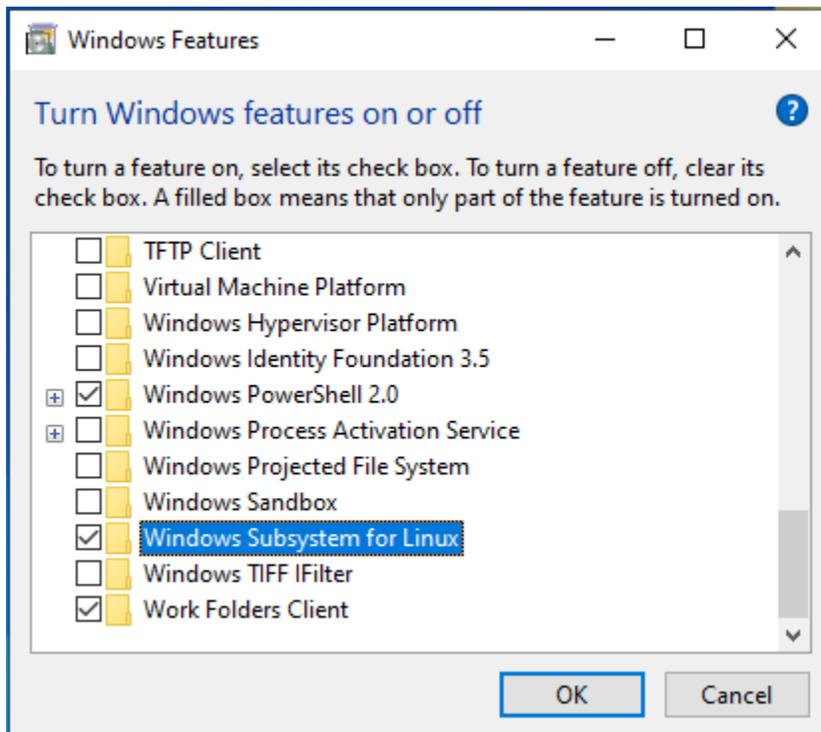
- File and Printer Sharing (Echo Request – ICMPv4-In) Private
- File and Printer Sharing (Echo Request – ICMPv6-In) Private

This will allow GNS3 to send back ICMP (ping) requests to the host PC.

2.4 WSL-1 Installation

For downloading and installation of Windows Subsystem for Linux on Windows 10 simply follow these steps:

- Step 1:- Enable the Windows Subsystem for Linux feature. This step needs administrative privileges.
 1. Right-click on the Windows Start menu icon, choose Search and type “Windows Features”. Select the top entry (Turn Windows features on or off) to enable or turn off Windows-Features. The Windows-Features dialog will be opened.
 2. Select in the upcoming dialog the check box for Windows Subsystem for Linux from the bottom of the list and press the OK button. Applying the changes may take a few minutes. Finally, press the Restart now button to reboot the computer.



Open the Start menu and select Microsoft Store and download Ubuntu (WSL). After installing WSL open WSL terminal and ping the above lab.

ANSIBLE INSTALLATION

Ansible handles communication between control nodes and managed nodes through multiple protocols:

- network_cli by SSH
- netconf by SSH
- httpapi by HTTP/HTTPS

Ansible can be installed on most Unix/Linux systems, with the only dependency being Python3.5 or higher. Currently, the Windows operating system is not officially supported as the control machine.

For installation, you can use the Ansible [website](#), which supports various kinds of platforms. We will be installing Ansible on the WSL (Ubuntu) machine.

```
sudo apt update
sudo apt-get install software-properties-common
sudo apt-add-repository ppa:ansible/ansible
sudo apt-get install ansible
```

Ansible is updated once every six months, so it's best to install it in a virtual environment.

```
sudo apt install virtualenv
virtualenv ansible-lab
source ansible-lab/bin/activate
pip install ansible==2.9.27
```

Stop the virtual environment with the `deactivate` command.

Install Ansible using your preferred method. See [Installing Ansible](#). Now we can check our managed node with the Ad-hoc command.

3.1 Ansible Ad-hoc Command

An Ansible Ad-hoc command uses the `/usr/bin/ansible` command-line tool to automate a single task on one or more managed nodes. Ad-hoc commands are quick and easy, but they are not reusable.

To run an Ad-hoc command or Ansible playbook it is mandatory to establish a manual SSH connection to the managed nodes to confirm your credentials and retrieve its configuration. This manual connection also establishes the authenticity of the network device, adding its RSA key fingerprint to your list of known hosts.

Instead of manually connecting and running a command on the network device, you can retrieve its configuration with a single, stripped-down Ansible command as below:

```
$ ansible all -i 9.9.9.100, -c network_cli -u admin -k -m ping -e ansible_network_os=ios
SSH password:
9.9.9.100 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
```

The flags in this command set seven values:

- the host group(s) to which the command should apply (in this case, all)
- the inventory (-i, the device or devices to target - without the trailing comma -i points to an inventory file)
- the connection method (-c, the method for connecting and executing ansible)
- the user (-u, the username for the SSH connection)
- the SSH connection method (-k, please prompt for the password)
- the module (-m, the ansible module to run)
- an extra variable (-e, in this case, setting the network OS value)

3.2 Ansible modules

A large number of modules are also installed alongside the installation of Ansible. Modules are responsible for the actions that Ansible executes on managed nodes. Furthermore, each module is responsible for a specific task. Modules can be run individually, in ad-hoc commands, or assembled into a particular job (play) and then into a playbook.

For example, we have already executed ad-hoc commands using the `ios_command` module and passed an argument.

```
ansible 9.9.9.101 -m ios_command -a "commands='sh ip int br'"
```

Ansible modules are generally idempotent. This means that the module can be executed as many times as desired, but the module will only make changes if the system is not in the desired state.

3.3 Ansible Raw module

The raw module is the module that doesn't translate our commands (not going through the module subsystem), Ansible purely transfer our commands on remote devices via SSH. A common case is installing python on a system without python installed by default or devices such as routers that do not have any Python installed. The raw module is mainly used for monitoring and troubleshooting.

```
$ ansible all -i 9.9.9.100, -u admin -m raw -a "sh ver" -k
SSH password:
9.9.9.100 | CHANGED | rc=0 >>
Cisco IOS Software, vios_l2 Software (vios_l2-ADVENTERPRISEK9-M), Version 15.2(CML_
↪ NIGHTLY_20180619)FLO_DSGS7, EARLY DEPLOYMENT DEVELOPMENT BUILD, synced to V152_6_0_81_
↪ E
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2018 by Cisco Systems, Inc.
```

(continues on next page)

(continued from previous page)

```
Compiled Tue 19-Jun-18 06:06 by mmen
```

```
ROM: Bootstrap program is IOSv
```

```
SW1 uptime is 28 minutes
```

```
System returned to ROM by reload
```

```
System image file is "flash0:/vios_l2-adventerprisek9-m"
```

```
Last reload reason: Unknown reason
```

This product contains cryptographic features and is subject to United States and local country laws governing import, export, transfer and use. Delivery of Cisco cryptographic products does not imply third-party authority to import, export, distribute or use encryption. Importers, exporters, distributors and users are responsible for compliance with U.S. and local country laws. By using this product you agree to comply with applicable laws and regulations. If you are unable to comply with U.S. and local laws, return this product immediately.

A summary of U.S. laws governing Cisco cryptographic products may be found at: <http://www.cisco.com/wwl/export/crypto/tool/stqrg.html>

If you require further assistance please contact us by sending email to export@cisco.com.

```
Cisco IOSv () processor (revision 1.0) with 574709K/209920K bytes of memory.
```

```
Processor board ID 9S6W017B82L
```

```
1 Virtual Ethernet interface
```

```
16 Gigabit Ethernet interfaces
```

```
DRAM configuration is 72 bits wide with parity disabled.
```

```
256K bytes of non-volatile configuration memory.
```

```
2097152K bytes of ATA System CompactFlash 0 (Read/Write)
```

```
0K bytes of ATA CompactFlash 1 (Read/Write)
```

```
0K bytes of ATA CompactFlash 2 (Read/Write)
```

```
0K bytes of ATA CompactFlash 3 (Read/Write)
```

```
Configuration register is 0x101
```

```
*****
* IOSv is strictly limited to use for evaluation, demonstration and IOS *
* education. IOSv is provided as-is and is not supported by Cisco's *
* Technical Advisory Center. Any use or disclosure, in whole or in part, *
* of the IOSv Software or Documentation to any third party for any *
* purposes is expressly prohibited except as otherwise authorized by *
* Cisco in writing. *
*****Shared_
```

```
↔connection to 9.9.9.100 closed.
```

We probably don't want to see every line of the output from "show version" command. So we can use grep, to filter, the name of the device and also the version.

```
$ ansible all -i 9.9.9.100, -u admin -m raw -a "sh ver" -k | grep "CHANGED\|Version"
↳
SSH password:
9.9.9.100 | CHANGED | rc=0 >>
Cisco IOS Software, vios_l2 Software (vios_l2-ADVENTERPRISEK9-M), Version 15.2(CML_
↳NIGHTLY_20180619)FLO_DSGS7, EARLY DEPLOYMENT DEVELOPMENT BUILD, synced to V152_6_0_81_
↳E
```

One more example:

```
ansible all -i 9.9.9.101,9.9.9.102 -u admin -m raw -a "sh ip int bri | ex unass" -k
SSH password:
9.9.9.101 | CHANGED | rc=0 >>

Interface                IP-Address      OK? Method Status      Protocol
FastEthernet0/0          9.9.9.101       YES NVRAM  up          up
Shared connection to 9.9.9.101 closed.

9.9.9.102 | CHANGED | rc=0 >>

Interface                IP-Address      OK? Method Status      Protocol
FastEthernet0/0          9.9.9.102       YES NVRAM  up          up
Shared connection to 9.9.9.102 closed.
```

More practical examples are below:

```
$ ansible all -m raw -u admin -a "show running-config | include username" -k
$ ansible all -m raw -u admin -a "show ip interface brief" -k
$ ansible all -m raw -u admin -a "show ip interface brief | exclude unass" -k
$ ansible all -m raw -u admin -a "show arp" -k
$ ansible all -m raw -u admin -a "show mac address-table | include " -k
```

ANSIBLE INVENTORY

Ansible is easy to get started with, but before we start using Ansible for network automation, there is a minimum requirement we need to prepare Ansible. That is preparing Ansible configuration and inventory files.

- inventory file - describes devices
- Ansible configuration file setting, to work with network equipment

4.1 Inventory file

In the inventory file we add the names or IP of the devices that are to be managed remotely. Devices can be listed one at a time or divided into groups. Preferably use the name of the devices instead of the IP addresses and make sure that these names can be resolved via the DNS server or from the control machine local `/etc/hosts` (static DNS entries) file.

The file can be described in INI or YAML format. Example files in INI format:

```
rtr.example.com

[router]
192.168.10.3
192.168.20.4

[switch]
192.168.30.5
```

The name, which is indicated in square brackets, is the name of the group. In this case, two device groups are created: that is router and switch.

The division into groups must be approached carefully. By default, the inventory file is located in `/etc/ansible/hosts`. In this case, it is usually better to create your own inventory file and use it. To do this, either specify it when starting ansible using the `-i path` option, or specify the file in the Ansible configuration file.

4.2 Group of groups

Ansible also allows you to combine groups of devices into a common group. A special syntax is used for this:

```
[router]
192.168.10.3
192.168.20.4

[switch]
192.168.30.5

[cisco:children]
router
switch
```

4.3 Default groups

There are two default groups all and ungrouped. The all group contains every host.

The ungrouped group contains all hosts that don't have another group aside from all. Every host will always belong to at least 2 groups (all and ungrouped or all and some other group). There are essential keys that are primarily used in the inventory file in Ansible.

- children to create groups of group
- hosts to add device to the group
- vars to define variables

By default, Ansible has two groups: all and ungrouped. The first includes all hosts, and the second, respectively, hosts that do not belong to any of the groups.

4.4 Adding Variables

We can store variable values that relate to a specific host or group in inventory file. You may add variables directly to the hosts and groups in your main inventory.

When working with network equipment, you must specify that a network_cli connection should be used. This can be specified in the inventory file, variable files, and so on.

To verify the hosts in your inventory:

```
$ ansible all --list-hosts
SSH password:
hosts (3):
  192.168.10.3
  192.168.20.4
  192.168.30.5
```

4.5 Build Inventory for Lab

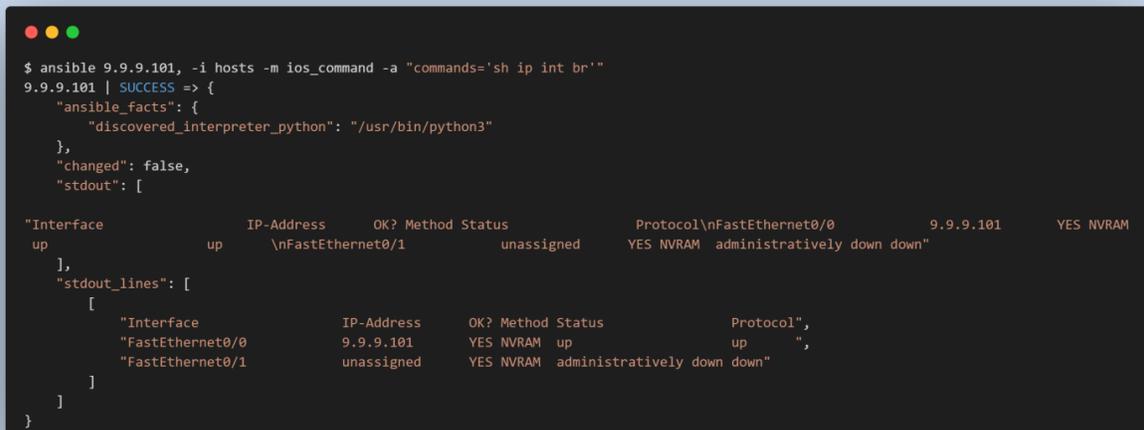
Now let's create an inventory hosts file. In the first step we enter the name of devices into group and then add groups to the network group as below:

```
[router]
9.9.9.101
9.9.9.102
9.9.9.103

[router:vars]
ansible_user=admin
ansible_password=cisco
ansible_network_os=ios
ansible_connection=network_cli
```

Some of the parameters (variables) can be written to the inventory file and then they will not need to be specified in the ad-hoc command.

Now the ad-hoc command can be called like in the below screenshot:



```
$ ansible 9.9.9.101, -i hosts -m ios_command -a "commands='sh ip int br'"
9.9.9.101 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "stdout": [

"Interface      IP-Address      OK? Method Status      Protocol\nFastEthernet0/0      9.9.9.101      YES NVRAM      administratively down down"
up
  ],
  "stdout_lines": [
    [
      "Interface      IP-Address      OK? Method Status      Protocol",
      "FastEthernet0/0      9.9.9.101      YES NVRAM      up          ",
      "FastEthernet0/1      unassigned      YES NVRAM      administratively down down"
    ]
  ]
}
```

4.6 Ansible Inventory in YAML

Our last inventory file was the INI format. There is also another format called .yaml or .yml which is more readable and may be required depending on the version of Ansible or module. Therefore, it is a good idea to learn how to create an inventory file in YAML format.

Another important point about the YAML inventory file is indentation. This means that all keys and values at the same level must have the same indentation. This is the only way to separate key/value pairs without parentheses or brackets. Let's first create the groups, add devices and then the variables.

In our previous inventory file, we have all group by default and router as sub-groups. So first we create router and switch sub groups, and then we add devices with the help of "hosts" key in sub-groups. Then lastly, we create all group add sub group with the help of the "children" and variables with the help of "vars" key.

Below is the result of the inventory file in the format of YAML.

```
---
router:
  hosts:
    R1:
      ansible_host: 9.9.9.101
    R2:
      ansible_host: 9.9.9.102
    R3:
      ansible_host: 9.9.9.103

switch:
  SW1:
    ansible_host: 9.9.9.100

cisco:
  children:
    router:
    switch:
  vars:
    ansible_network_os: ios
    ansible_connection: network_cli
```

You can read more about How to build your inventory on the [website](#) documentation.

4.6.1 Inventory Checking Commands

```
ansible-inventory --list -i <hosts.ini>
ansible-inventory --graph
ansible-inventory --host sw1 -i <inventory.yml>
```

ANSIBLE CONFIGURATION

Ansible settings can be changed in the configuration file. Ansible supports several sources for configuring its behavior including command line option; any command-line option will override any configuration setting. The Ansible configuration file can be stored in different locations (Ansible will process the below list and use the first file found, all other settings are ignored.). See [precedence rules](#) for more information.

- ANSIBLE_CONFIG environment variable
- ansible.cfg in the current directory
- ~/.ansible.cfg in the user's home directory
- /etc/ansible/ansible.cfg default location

Ansible looks for the configuration file in the specified order and uses the first one it finds. But user can change many parameters in the configuration file. A complete list of parameters and their descriptions can be found in the [documentation](#). In the current directory, create the following ansible.cfg configuration file:

```
[defaults]
inventory = ./hosts
remote_user = admin
ask_pass = True
```

Parameter settings in the configuration file:

- [defaults] - This configuration section describes the general default settings.
- inventory = ./hosts - The inventory allows you to specify the location of the inventory file.
- remote_user = cisco - The user Ansible will connect to managed node.
- ask_pass = True - this option is the same as the `--ask-pass` option on the command line.

Now we have set parameters in our configuration file, so we need to amend the variable in the inventory file as below:

```
[router:vars]
ansible_network_os=ios
ansible_connection=network_cli
```

When we use the `ansible --version` command, it will show the exact location of the configuration file, which by default is in `/etc/ansible/ansible.cfg`, first line shows the location of Ansible configuration file.

```
$ ansible --version
ansible 2.9.27
  config file = /home/syed/ansible.cfg
  configured module search path = ['/home/syed/.ansible/plugins/modules', '/usr/share/
↪ansible/plugins/modules']
```

(continues on next page)

(continued from previous page)

```
ansible python module location = /home/syed/.local/lib/python3.10/site-packages/ansible
executable location = /home/syed/.local/bin/ansible
python version = 3.10.6 (main, Aug 10 2022, 11:40:04) [GCC 11.3.0]
```

Fortunately, we can override both default Ansible configuration file and inventory file for each project, since each project may have its own configuration parameters and a different list of devices to be managed remotely. It is therefore recommended to create a folder with its own configuration and inventory file for each project.

At this stage, if we run our previous ad-hoc command the result will be the same except for asking for the password.

5.1 Gathering Facts

Ansible By default collects facts about devices; these facts are information about the hosts Ansible connects to and can be used as variables in playbooks/templates.

By default, facts are collected by the setup module. This module is automatically triggered by playbooks when run, to gather useful variables about remote hosts that can be used in playbooks.

The setup module is not suitable for network equipment, so the collection of facts must be disabled. This can be done in the Ansible configuration file or in the playbook. Disabling fact gathering in the configuration file:

```
gathering = explicit
```

5.2 Host Key Checking

The host key checking parameter in the configuration file is responsible for checking keys when connecting via SSH. If you specify `host_key_checking = False`, keys checking will be disabled. This is useful when the Ansible control node needs to connect with a large number of devices for the first time.

Ansible configuration file can be checked with the below commands.

```
ansible-config view
ansible-config dump --only-changed
```