Introduction to Ansible

Why Automation?

- **Tasks in code**
- **Collaboration**
- **Eliminate errors**
- **Write once**
- **Lazines**
- **Etc….**

Possible Used Cases Includes

- **Configuration Management**
- **Maintenance**
- **Configuration Audit**
- **Upgradation**

Configuration Management

Lets say an ISP needs to configure 100s or 1000s of vlans on n number of devices to accommodate new customers

A customer needs to deploy a new data center or new site which consists large number of devices and variety of vendors.

Legacy Solution

Prepare configuration for all the devices.

Login to the devices one by one and configure it.

Think about the time it is going to take when we prepare the configuration of the devices one by one.

Then login to the devices one by one to configure them.

This could be error prone.

Ansible Solution

We can use Ansible module to generate the configuration and to configure the devices as well.

Power Maintenance/ Power Issue

- **Devices went down and came back up.**
- **How to confirm**
  - **whether all the devices has come back online.**
  - **whether the connectivity has restored or not.**

Legacy Solution

Login to all the devices one by one and run the set of command one by one.

- show runn
- show version
- show int status
- show ip ospf neighbor
- show ip bgp neighbor

Ansible Solution

We can ping a set of devices with Ansible and can fetch the required information like

- Fetching Configuration before and after the maintenance.
- uptime of all the devices
- routing information
- link status

All the commands used in legacy solution can be run in one go with Ansible on a set of devices.

ISP Maintenance/ Flap

- **Service restoration**
- **Routing Neighborship, Routes**
- **Interface status**

Legacy Solution

Login to all the devices one by one and run the set of command one by one.

```
ping 8.8.8.8
show int status
show ip ospf neighbor
show ip bgp neighbor
show runn
show version
```

Ansible Solution

We can ping a set of devices with Ansible and can fetch the required information like

- Connectivity with Internet & Intranet
- uptime of all the devices
- routing information
- link status

All the commands used in legacy solution can be run in one go with Ansible on a set of devices.

Some other use cases can be

Configuration Audit

OS Upgradation

Legacy Solution

Perform the tasks on individual devices.

Ansible Solution

Perform the tasks on multiple devices.

Installation Requirements

The OS that works well with Ansible are

Fedora

Linux 7

CentOS

CentOS 7 → http://isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7-x86_64-Everything-1804.iso

Work around for Windows Users.

Microsoft Windows as a control machine is not supported.

Windows users need to install either VMware workstation or Oracle VM Virtual Box. VMware has license issues so, Oracle VM Virtual Box is best.

Oracle VM Virtual Box → https://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html

Python 2.6 or later/Python 3

We also need a text editor.

VIM

Notepad++ → https://notepad-plus-plus.org/download/v7.6.html

Sublime → https://www.sublimetext.com/

Tools for Network Engineers

We have few tools to practice Ansible from networking perspective. GNS3 & EVE.

Use following link for GNS3 →

https://www.youtube.com/watch?v=x9pGYyEqLYs&list=PLhfrWIlLOoKNFP_e5xcx5e2GDJIgk3ep6

GNS3 supports a variety of vendors. It supports Cisco (Routers, Switches, WLC, Prime), Palo Alto, F5, Juniper and many others. We can use ansible with all the supported vendors.

Use following link for EVE-NG →

http://www.eve-ng.net/

Ansible Introduction

Ansible, at its core, is a task execution engine.

It provides a method to easily define one or more actions to be performed on one or more Servers/Network Devices. We don't need to login to the device. These tasks can target the local system Ansible is running from, as well as other systems Ansible can reach over the network.

Open Source hosted on GitHub written in Python. User interaction is YMAL.

Github is a code repository, which can be integrated to other tools like Jenkins to develop CI/CD model.

YAML works as a key pair value. It`s user friendly and human readable.

Ansible is the most suitable tools for Network Automation.

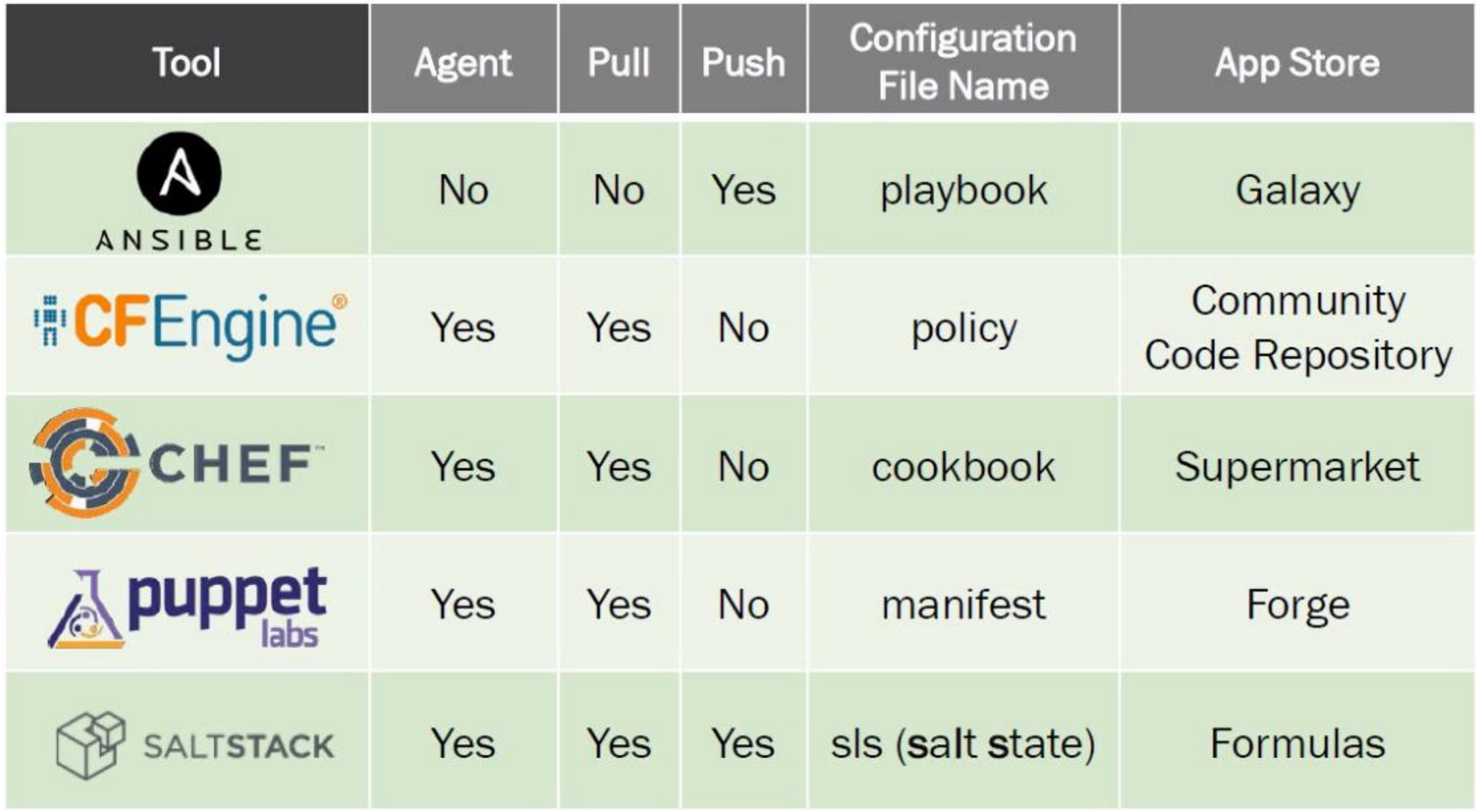
It works on SSH and most of the network devices are accessible over SSH.

It does not need any agent to be installed on the remote device. We will not be able to install agent on most of the network devices.

As an agent can`t be installed, so, we can`t use a tool which works on pull mechanism. We need a tool which works on push mechanism. Ansible works on push mechanism.

The only requirement is the generic username and password. There might be some security related issues with generic username and password. Now we have CyberArk module which can help us to resolve this.

Ansible and Other Automation Tolls



Tool	Agent	Pull	Push	Configuration File Name	App Store
ANSIBLE	No	No	Yes	playbook	Galaxy
CFEngine	Yes	Yes	No	policy	Community Code Repository
CHEF	Yes	Yes	No	cookbook	Supermarket
puppet labs	Yes	Yes	No	manifest	Forge
SALTSTACK	Yes	Yes	Yes	sls (salt state)	Formulas

Installation and Verification

Follow the link to install Ansible on different OS ->
https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

Follow the following steps to install Ansible in CentOS7

```
yum update
yum install epel-release
yum install python-pip
pip install --upgrade pip
yum install ansible
```

Alternatively, Ansible can be installed via Python pip in a Python virtual environment. This is useful if I want to make use of multiple versions of Ansible or if I'm operating in an environment where I do not have rights to install things in the system path.

If we are using GNS3 to practice Ansible with networking, then we do not need to install Ansible. Its pre-installed. We just need to configure it.

If we are using eve-ng then we need to install linux and install Ansible.

Once the Ansible is installed successfully, use following commands to verify:

which ansible -> check the availability of Ansible

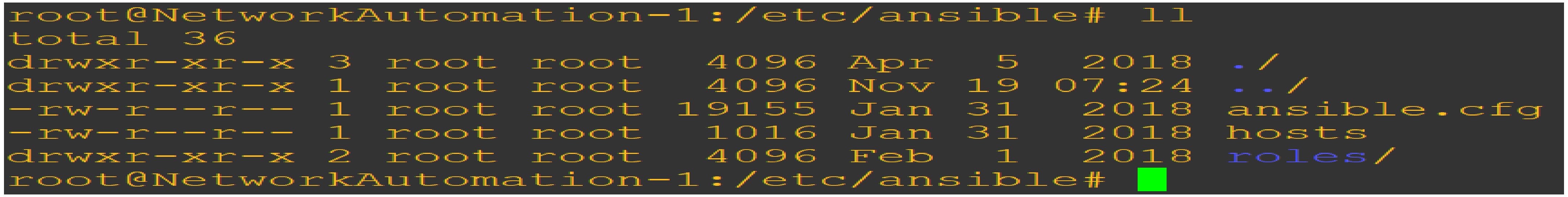
version --ansible -> check which version is installed

Version

configuration file in use at etc/ansible/ansible/cfg

module search path is the default.

Once Ansible is installed go to cd /etc/ansble/ and run ll/ls it will show all the files and folders.



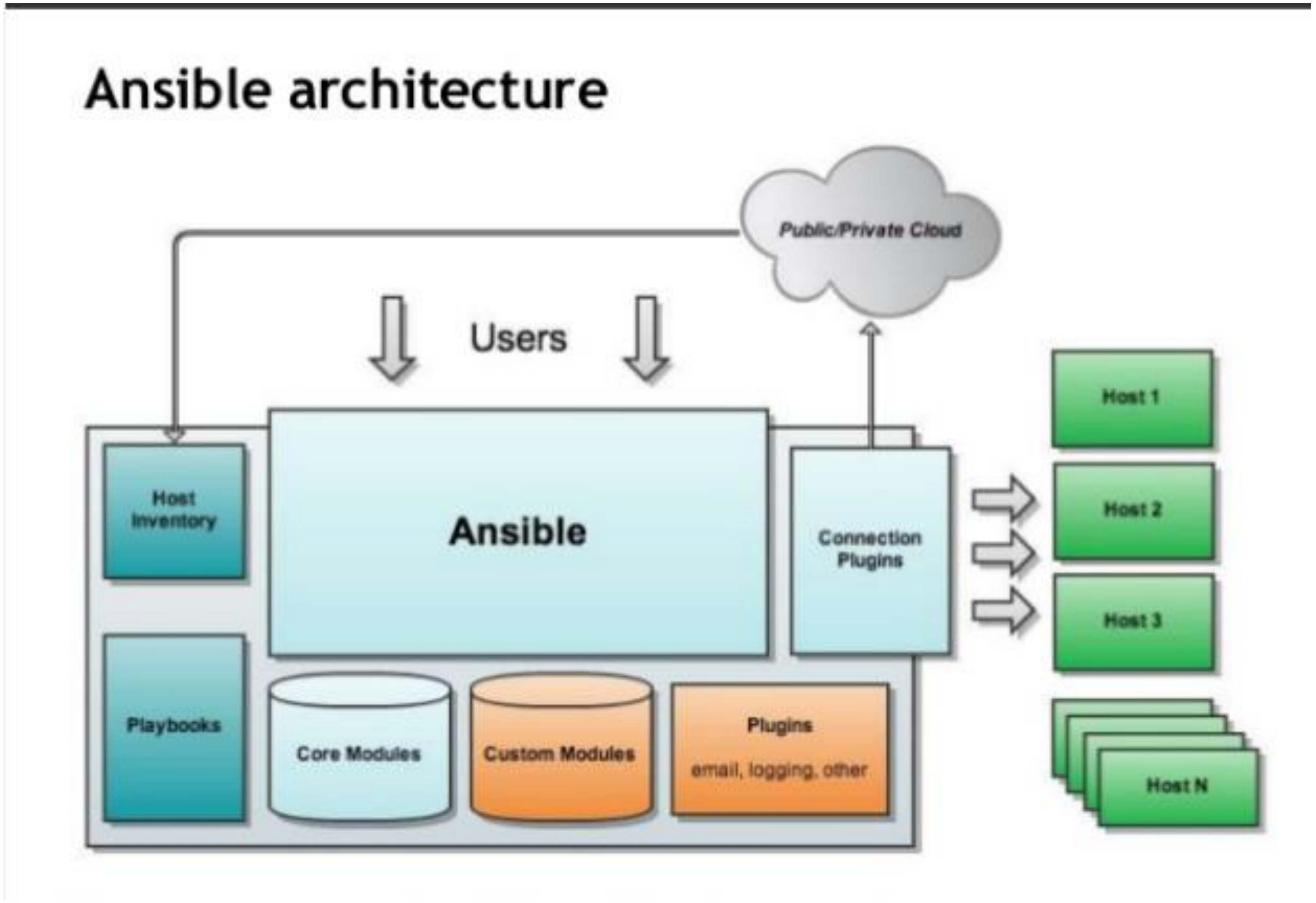
```
root@NetworkAutomation-1:/etc/ansible# ll
total 36
drwxr-xr-x 3 root root  4096 Apr  5  2018 ./
drwxr-xr-x 1 root root  4096 Nov 19 07:24 ../
-rw-r--r-- 1 root root 19155 Jan 31  2018 ansible.cfg
-rw-r--r-- 1 root root  1016 Jan 31  2018 hosts
drwxr-xr-x 2 root root  4096 Feb  1  2018 roles/
root@NetworkAutomation-1:/etc/ansible#
```

Ansible.cfg → Certain settings in Ansible are adjustable via a configuration file (ansible.cfg).

Hosts → Ansible works against multiple systems in your infrastructure at the same time. It does this by selecting portions of systems listed in Ansible's inventory, which defaults to being saved in the location **/etc/ansible/hosts**. You can specify a different inventory file using the **-i <path>** option on the command line.

Roles → In Ansible, Role is a primary mechanism for breaking a playbook into multiple files.

Ansible Architecture



A brief explanation of Ansible and Its parts

Ease of getting started

- An inventory of targets – hosts file with target hosts (network devices)
- A state to assert – tasks (fetching information or configuration)
- Credential to log into a device – works on SSH which is already open in the network

Inventory

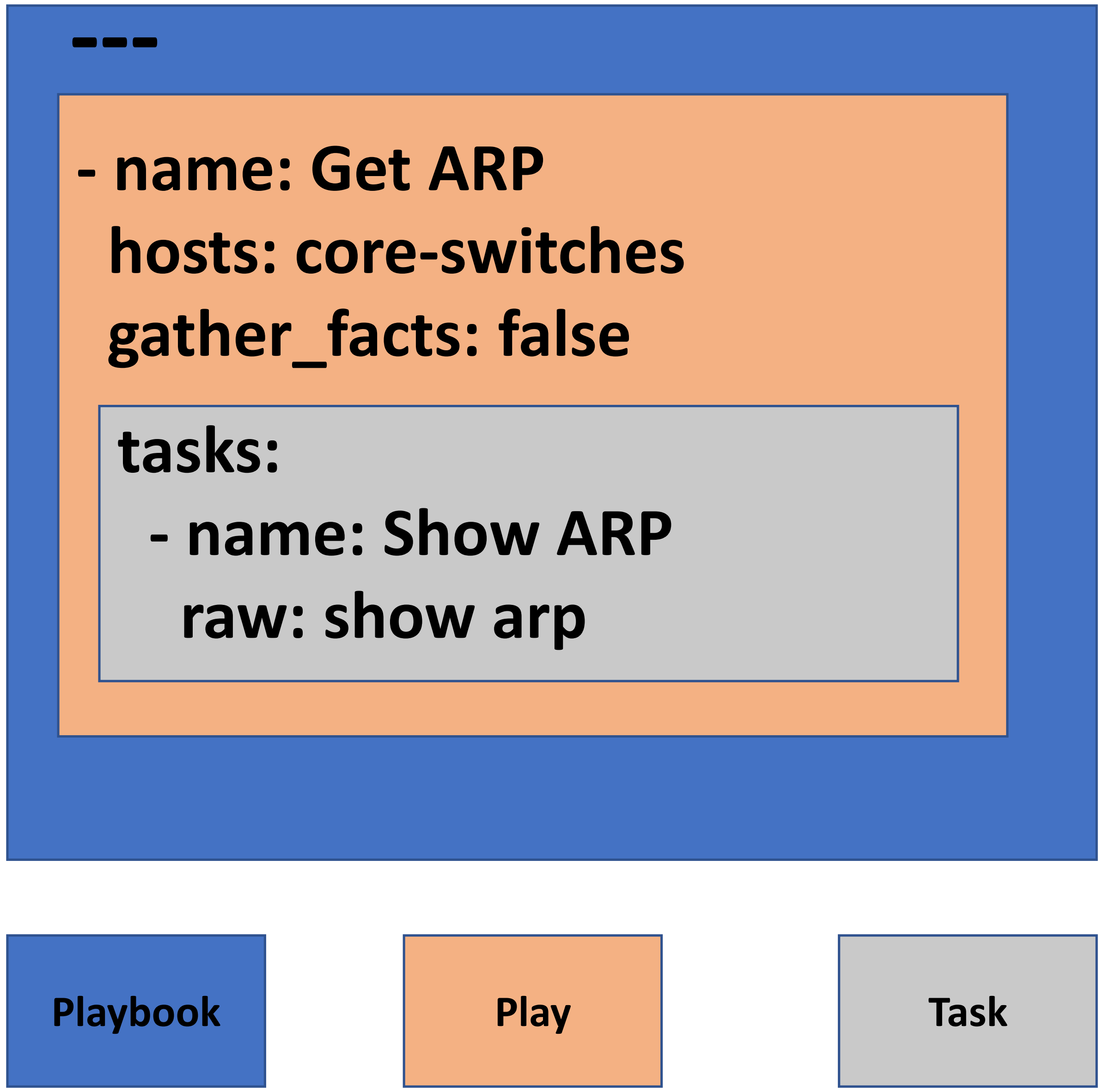
A set of potential target hosts to execute tasks on. Inventories are not specifically tied to a set of Ansible instructions. Multiple inventories can exist and be used at execution time.

```
as-01
as-02
cs-01
cs-02
```

```
[access-switches]
as-01
as-02
[core-switches]
cs-01
cs-02
```

Hosts or groups are used in patterns as an entity to target or as an entity to skip from within a target. Patterns support wild cards and even regular expressions.

Playbook and Its Components



```
---
- name: Get ARP
  hosts: core-switches
  gather_facts: false
  tasks:
    - name: Show ARP
      raw: show arp
```

Control Node

Any machine with Ansible installed. You can run commands and playbooks, invoking /usr/bin/ansible or /usr/bin/ansible-playbook, from any control node. You can use any computer that has Python installed on it as a control node - laptops, shared desktops, and servers can all run Ansible. However, you cannot use a Windows machine as a control node. You can have multiple control nodes.

Managed Nodes

The network devices (and/or servers) you manage with Ansible. Managed nodes are also sometimes called “hosts”. Ansible is not installed on managed nodes.

Inventory

A list of managed nodes. An inventory file is also sometimes called a “hostfile”. Your inventory can specify information like IP address for each managed node. An inventory can also organize managed nodes, creating and nesting groups for easier scaling.

Modules

The units of code Ansible executes. 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. You can invoke a single module with a task, or invoke several different modules in a playbook.

Tasks

The units of action in Ansible. You can execute a single task once with an ad-hoc command.

Playbooks

Ordered lists of tasks, saved so you can run those tasks in that order repeatedly. Playbooks can include variables as well as tasks. Playbooks are written in YAML and are easy to read, write, share and understand.

Lets see a sample Playbook

$ ***nano show-runn.ymal***

```
---                                   → indicates the start of yaml file
- name: Show Running Configuration    → name of the play
  hosts: access-switches              → inventory - hosts
  connection: local                   → ansible itself will not connect to hosts
  gather_facts: false                 → ansible collects some info from hosts which we can use later

  tasks:                              →it includes module and set of commands that will ne run
  - name: Show Runn                   → name of the task
    ios_command:                      → ansible module → see ansible docs
      authorize: yes                  → enable mode
      commands:                       → IOS based commands will be listed under it.
        - show runn
    register: print_output            → registering output of the command
  - debug: var=print_output.stdout_lines      → ansible module displaying output on screen
```

Run playbook

$ ansible-playbook show-runn.yml -u username -k (it will ask for password)

Ad-hoc

- Ad-Hoc commands and Play-Books both can be used to fetch information from network devices and for troubleshooting purpose.
- Ad-hoc commands are very useful and handy for troubleshooting and pulling out configurations or output of a command and then saving it to some files for future purpose.
- Ad-hoc commands are like IOS commands with some more arguments running directly on the devices, while ad-hoc command runs from the Ansible server, the only difference is the syntax.

Examples of Hd-hoc commands

```
ansible all -m raw -a "show version" -u cisco -k | grep SUCCESS
SSH password:
Core1 | SUCCESS | rc=0 >>
Core2 | SUCCESS | rc=0 >>
```

```
ansible all -m raw -a "show version" -u cisco -k | grep 'SUCCESS\|UNREACHABLE'
SSH password:
Core2 | SUCCESS | rc=0 >>
Core1 | SUCCESS | rc=0 >>

Access1 | UNREACHABLE! => {
Access2 | UNREACHABLE! => {
Access3 | UNREACHABLE! => {
```