**ANSIBLE**

**ANSIBLE is a configuration management tool**

**What is Configuration management?**

[Configuration management](https://www.netapp.com/devops-solutions/configuration-management) (CM) is an automated method for maintaining computer systems and software in a known, desired state.

**Automation History**

1. Bash Scripting / Batch Scripting
2. Python / Ruby
3. PowerShell
4. Puppet
5. Salt Stack
6. Chef
7. Ansible

**What is Ansible**

Ansible is an open-source automation tool, or platform, used for IT tasks such as [configuration management](https://www.simplilearn.com/configuration-management-2-article), application deployment, intraservice orchestration, and provisioning.

**Benefits of Ansible**

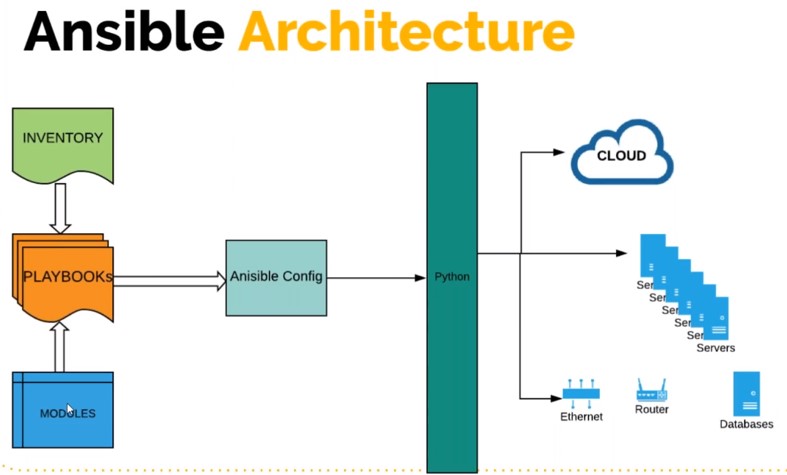
* **Simple:** Ansible is very simple tool.
* **Free:** Ansible is an open-source tool.
* **Very simple to set up and use:** No special [coding](https://www.simplilearn.com/tutorials/programming-tutorial/coding-for-beginners) skills are necessary to use Ansible’s playbooks
* **Powerful:** Ansible lets you model even highly complex IT workflows.
* **Flexible:**You can orchestrate the entire application environment no matter where it’s deployed. You can also customize it based on your needs.
* **Agentless:** You don’t need to install any other software or firewall ports on the client systems you want to automate. You also don’t have to set up a separate management structure.
* **YAML:** No Programming structured, its easy to read and write.

**Use cases of Ansible**

****

* Automation (Any system automation, Server, Database, configuration, start restart services)
* Change Management (Production server changes)
* Provisioning (Setup server from scratch or cloud provisioning)
* Orchestration (Large scale automation framework, can integrate with other tool like Jenkins, docker)

**Ansible Architecture**

****

### **What is Ansible Playbook**

Playbooks are the files where the Ansible code is written. Playbooks are written in YAML format. YAML means "Yet Another Markup Language,". It is basically a blueprint of automation tasks—which are complex IT actions executed with limited or no human involvement.

### **What is Ansible Inventory**

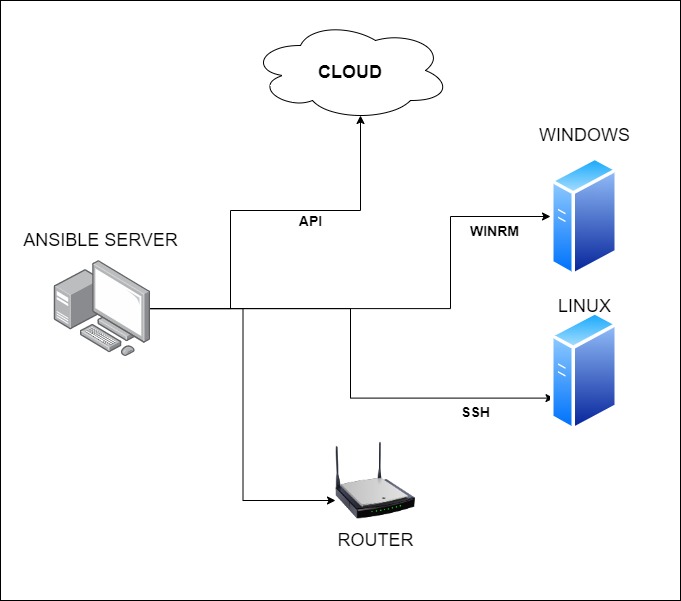
Inventory file defines the hosts and groups of hosts upon which commands, modules, and tasks in a playbook operate. The file can be in one of many formats depending on your Ansible environment and plugins. The default inventory located at **/etc/ansible/hosts**

### **What is Ansible Config file**

The file that governs the behavior of all interactions performed by the control node. In Ansible’s case that default configuration file is (ansible.cfg) located in /etc/ansible/ansible.cfg.

### Ansible uses the python module, python script to connect to the target machine. It dumps the python script and execute there and return the output.

**Ansible Connections:**

****

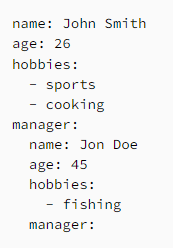
**Control machine**: A control machine is the central node in an Ansible infrastructure. It is used to manage all the other machines in the network.

**Remote machine**: A remote machine is any machine that is not the control machine. Remote machines are managed by the control machine using SSH.

**Target machine**: A target machine is a remote machine being provisioned or configured by Ansible.

**JSON and YAML**

**JSON YAML**



[**https://docs.ansible.com/ansible/latest/reference\_appendices/YAMLSyntax.html**](https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html)

**ANSIBLE SETUP on AWS Cloud**

[**https://docs.ansible.com/ansible/latest/installation\_guide/installation\_distros.html**](https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html)

* Launch EC2 instance for Control Machine
  + Give a name as Control Machine
  + Select Ubuntu 18.04 image
  + Create a security group as “control-sg”
  + Create a Key pair as “control-key”
  + Copy and paste the Setup.sh script into User-data field
  + Launch the instance
* Let’s Launch EC2 instance for Target machine
  + Give a name as “Centos-target”
  + Select Centos 7 image from Marketplace
  + Keep t2.micro instance type
  + Give count as 2 machines
  + Create security group – webapp-sg
    - Allow port 22 from myip
    - Allow port 22 from “control-sg”
  + Create new key pair for these webapp instances.
* Let’s launch one more EC2 instance for database
  + Give a name as “db01”
  + Select Centos 7 image from Marketplace
  + Keep t2.micro instance type
  + Give count as 1 only
  + Create a key pair
  + Create a security group as “db-sg”
    - Allow port 22 from myip
    - Allow port 22 from “control-sg”

Let’s login to Control Machine

* Check Ansible version: - $ ansible –version
* Create a project folder: - $ mkdir sscademy
* Create exercises inside this dir:- $ mkdir exercise1

**Inventory and Ping Module**

Default/Global inventory location

**$ ls /etc/ansible/hosts**

**We can create our own inventory file inside the project directory. Its always best practice to create project specific inventory file.**

**$ cd exercise1**

**$ vim inventory**

web01 ansible\_host=private\_ip\_of\_web01 ansible\_user=centos ansible\_ssh\_private\_key\_file=sscadmy-key.pem

web02 ansible\_host=private\_ip\_of\_web02 ansible\_user=centos ansible\_ssh\_private\_key\_file=sscadmy-key.pem

db01 ansible\_host=private\_ip\_of\_db01 ansible\_user=centos ansible\_ssh\_private\_key\_file=sscadmy-key.pem

:wq

**Create a ssh\_key.pem file**

$ vim sscademy-key.pem

Copy and paste the .pem key content inside this

$ chmod 400 sscademy-key.pem

**Let’s test it**

$ ansible -i inventory -m ping web01

Now it will ask for the host path confirmation (yes/no).

Instead of supplying yes or no, we can handle it in better way by editing the **ansible.cfg** file

$ sudo vim /etc/ansible/ansible.cfg

Find an entry called “**host\_key\_checking**”

Make it uncommented, **host\_key\_checking: False**

**Save and quit**

**=====================================================**

Now run ansible ad-hoc command again.

$ ansible -i inventory -m ping web01

$ ansible -i inventory -m ping web02

$ ansible -i inventory -m ping db01

Instead of pinging individual servers, we can make server group in the Inventory file.

$ vim inventory

web01 ansible\_host=private\_ip\_of\_web01 ansible\_user=centos ansible\_ssh\_private\_key\_file=sscadmy-key.pem

web02 ansible\_host=private\_ip\_of\_web02 ansible\_user=centos ansible\_ssh\_private\_key\_file=sscadmy-key.pem

db01 ansible\_host=private\_ip\_of\_db01 ansible\_user=centos ansible\_ssh\_private\_key\_file=sscadmy-key.pem

[webservergrp]

web01

web02

[dbservergrp]

db01

[dc\_nverginia:children]

webservergrp

dbservergrp

Now run ansible ad-hoc command again.

$ ansible -i inventory -m ping webservergrp

$ ansible -i inventory -m ping dbservergrp

$ ansible -i inventory -m ping dc\_nverginia

$ ansible -i inventory -m ping all

$ ansible -i inventory -m ping ‘\*’

$ vim inventory

web01 ansible\_host=private\_ip\_of\_web01

web02 ansible\_host=private\_ip\_of\_web02

db01 ansible\_host=private\_ip\_of\_db01 ansible\_user=centos ansible\_ssh\_private\_key\_file=sscadmy-key.pem

[webservergrp]

web01

web02

[dbservergrp]

db01

[dc\_nverginia:children]

webservergrp

dbservergrp

[webservergrp:vars]

ansible\_user=centos

ansible\_ssh\_private\_key\_file=sscadmy-key.pem

[dc\_nverginia:children]

ansible\_user=centos

ansible\_ssh\_private\_key\_file=sscadmy-key.pem

Test now

$ ansible -i inventory -m ping all

**Ad Hoc Commands**

$ cp -r exercise1/ exercise2

$ cd exercise2

**Commands:**

**Example 1:**

$ ansible -i inventory -m yum -a “name=httpd state=present” web01

$ ansible -i inventory -m yum -a “name=httpd state=present” web01 –become

*Run same command and check what will happen*

**Example 2:**

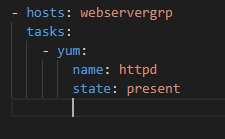
$ ansible -i inventory -m service -a “name=httpd state=started enabled=yes” web01 –become

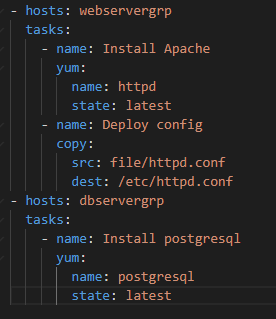
**Example 3:**

$ vim index.html

$ ansible -i inventory -m copy -a “src=index.html dest=/var/www/html/index.html” web01 –become

**ANSIBLE PLAYBOOKS written in YAML**





**Now lets start writing the Ansible playbooks**

* + Refer 2nd\_playbook.yaml
  + After writing the playbooks if you want to check the syntax then you can use the command
    - $ ansible-playbook -i inventory 2\_playbook.yaml –syntax-check
    - $ ansible-playbook -i inventory 2\_playbook.yaml

**ANSIBLE MODULES**

List all the ansible modules

$ ansible-doc -l

$ ansible-doc yum

<https://docs.ansible.com/ansible/2.9/modules/modules_by_category.html>

$ ansible-playbook -i inventory 2\_playbook.yaml -C

We can test the playbook before it actually applied. We call this as dry run.

Run 4\_db.yaml playbook to understand how to install mysql, add database

Now you will get an error for python.

Lets login to database server using ssh

$ yum search python | grep -i mysql

Copy the package name and we are going to add in the playbook

Now you can refer 5\_db.yaml playbook.

Execute the playbook again.

$ ansible-playbook -i inventory 5\_db.yaml

Now lets add a database user

Refer 6\_db.yaml playbook

$ ansible-playbook -i inventory 6\_db.yaml

**Ansible Configuration Settings**

**Order of Ansible Config**

1. **ANSIBLE\_CONFIG (environment variable if set)**
2. **ansible.cfg (in the current directory)**
3. **~/. ansible.cfg (in the home directory)**
4. **/etc/ansible/ansible.cfg**

Let’s explore the global ansible.cfg from **/etc/ansible/ansible.cfg**

After exploring the global configuration, lets create local ansible configuration in the project directory

**$ /ACM/exercise4/vi ansible.cfg**

[defaults]

host\_key\_checking = False

inventory = ./inventory

forks = 5

log\_path = /var/log/ansible.log

[privilege\_escalation]

become = True

become\_method = sudo

become\_ask\_pass = False

Let’s test the ansible using this configuration

$ ansible-playbook 4\_db.yaml

$ sudo touch /var/log/ansible.log

$ sudo chown ubuntu.ubuntu /var/log/ansible.log

$ ansible-playbook 4\_db.yaml

$ cat /var/log/ansible.log

$ ansible-playbook 4\_db.yaml –vv

--vv – verbosity

**Ansible Variables**

1. Playbook Variables
2. Inventory Based Variables

* In Inventory File
* group\_vars/all
* group\_vars/groupname
* host\_vars/hostname

1. Roles

1. **Playbook variables**

Let’s refer the variables defines inside the playbook.

**7\_db\_with\_vars.yaml**

**8\_db\_with\_vars\_debug.yaml**

1. **Inventory Based Variables**
   1. Variables for Groups –

**group\_vars/all**

**group\_vars/groupname**

* 1. Variables for hosts –

**host\_vars/hostname**

1. **Variables for Groups**

$ cp -r exercise6/\* exercise7

$ cd exercise7

$ mkdir group\_vars

$ vim group\_vars/all

# Variables accessible by ALL the Hosts from the Inventory file

dbuser: devopsadmin

dbpass: test123

dbname: devops

$ ansible-playbook 9\_db\_with\_vars\_from\_groupvars.yaml

1. **Variables for Host**

$ cp -r exercise7/\* exercise8

$ cd exercise8

$ rm -rf db.yaml index.html group\_vars/all

Refer 10\_vars\_precedence.yaml

Now let’s create **group\_vars/all** file

$ mkdir group\_vars

$ vim group\_vars/all

USERNM: globaluser

COMMNT: variable from group\_vars/all

**Now we can see that the precedence of the variables goes to variables mentioned in the playbook. If we comment the variables from the playbook, then it will take from group\_vars/all.**

Let’s work on **group\_vars/groupname**

As we have given the target machines group names in the inventory file, we can create the groupname file with same name.

**E.g. $ vi group\_vars/webservergrp**

USERNM: webservergrpuser

COMMNT: variable from group\_vars/webserver file

Let’s work on **host\_vars/hostname**

As we have given the target machines names in the inventory file, we can create the hostname file with same name.

E.g. $ mkdir host\_vars

$ vi host\_vars/hostname

$ vi host\_vars/web02

USERNM: web02user

COMMNT: variable from host\_vars/web02 file

So you have seen the highest priority goes first to playbook 🡪 host\_vars/hostname🡪 group\_vars/groupname 🡪 group\_vars/all

One more thing which super seeds all the priorities, which is variables passed through command line

$ ansible-playbook -e USERNM=cliuser -e COMMNT=cli vars\_precendence.yaml

**FACT VARIABLES: Setup Module**

Fact variables are run time variables they get generated when setup module got executed.

It will generate variables, some of them are as below

**gath ansible\_devices ansible\_default\_ipv4**

*OS names like RedHat, Debian etc Connected Device Information IP, MAC address, gateway etc*

**ansible\_processor\_cores ansible\_architecture**

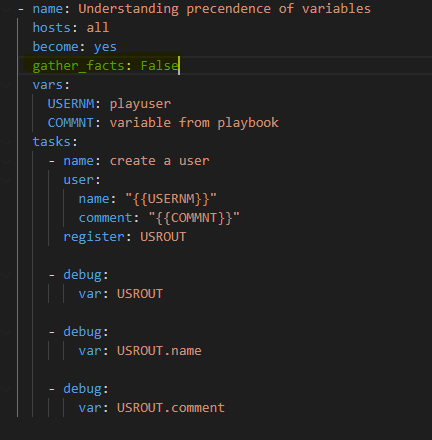
*Number of CPU cores 64-bit or 32-bit*

**ansible\_kernel**

Kernel Version

Now let’s login to Control Machine and learn about Fact Variables

* Execute any playbook, you will see “**Gathering Facts**”
  + Gathering Facts task will execute by default every time when we run the playbook.
  + It will gather the facts about the host which you have mentioned in your playbook
  + It uses a module called **setup**
* Let’s execute command for setup module to understand the gathering facts.
  + $ ansible -m setup web01
* You can disable the gathering facts option



Run the playbook – **12\_print\_fact.yaml** to understand how to print the fact variables.

Let’s Launch EC2 instance for Ubuntu operating system

* Login to AWS cloud
* Select Ubuntu 18.04
* Give name as “web03”
* Add in the same webserver security group
* Use the same .pem key which we have used for webapp
* Launch it
* Update our inventory file with new machine

**Decision Making & Loops**

**Decision Making using conditions**

Create a new exercise folder and will see these options

$ mkdir exercise10

$ cp -r exercise9/ exercise10

$ cd exercise10

$ vi 14\_configuring\_server.yaml

$ ansible -m setup web01 | grep ansible\_distribution

<https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_conditionals.html>

$ ansible-playbook 14\_configuring\_server.yaml

**LOOPS**

Refer **15\_loop.yaml** & **16\_add\_user\_with\_loop.yaml** playbooks to understand loops in Ansible

<https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_loops.html>

Let’s call the users from group\_vars/all file instead of hard coding in the playbook

$ mkdir group\_vars

$ vi group\_vars/all

usernames:

- user11

- user12

- user13

- user14

**FILE, COPY & TEMPLATE MODULE**

<https://docs.ansible.com/ansible/2.8/modules/list_of_files_modules.html>

$ mkdir exercise11

$ cp -r exercise10/\* exercise11

$ cd exercise11

Refer **18\_copy\_module.yaml** ansible playbook to understand how copy module works in Ansible playbook.

$ ansible-playbook 18\_copy\_module.yaml

Now banner file has been created, we will login to any machine and check it

$ ssh -i ssh-key.pem centos@private\_ip\_of\_server

Here you will see the banner

As we have install NTP service on our target machines. The NTP service configurations will be in the location **/etc/ntp.conf**. We will change the configuration of this ntp.conf file content through ansible playbook.

Login to centos machine and copy the entire ntp.conf content and come to control machine and create a file like

For RedHat family

$ vim ntp\_redhat.conf

Paste the content which we have copied from centos machine

Modify it with NTP server, for example consider these machines are in India

<https://www.pool.ntp.org/zone/in>

Consider if the server is in North America

<https://www.pool.ntp.org/zone/north-america>

Copy the NTP server details and change the server information in **ntp\_redhat.conf**

E.g.:

server 0.north-america.pool.ntp.org

server 1.north-america.pool.ntp.org

server 2.north-america.pool.ntp.org

server 3.north-america.pool.ntp.org

Now lets login to Ubuntu machine and copy the ntp.conf content, come back to control machine and create a conf file like

$ vi ntp\_debian.conf

Paste the content and edit the NTP server details & save it

Lets use template module to execute this changes.

$ mkdir templates

$ mv ntp\_\* templates/

$ cd templates

$ mv ntp\_redhat.conf ntp\_redhat.conf.j2

$ mv ntp\_debian.conf ntp\_debian.conf.j2

$ cd ..

Refer 19\_tempate\_module.yaml

These templates files are having static data, we can convert it to dynamic data

$ vi group\_vars/all

ntp0: 0.north-america.pool.ntp.org

ntp1: 1.north-america.pool.ntp.org

ntp2: 2.north-america.pool.ntp.org

ntp3: 3.north-america.pool.ntp.org

Now instead of hard coding the values in jinja2 templates files we can give using variables

0.north-america.pool.ntp.org => {{ntp0}}

1.north-america.pool.ntp.org => {{ntp1}}

2.north-america.pool.ntp.org => {{ntp2}}

3.north-america.pool.ntp.org => {{ntp3}}

$ vi templates/redhat.conf.j2

Edit and modify it as mentioned above.

$ ansible-playbook 19\_template\_module.yaml –syntax-check

$ ansible-playbook 19\_template\_module.yaml -C

$ ansible-playbook 19\_template\_module.yaml

**FILE MODULE**

Refer **20\_file\_module.yaml** playbook

<https://docs.ansible.com/ansible/2.8/modules/file_module.html#file-module>

**Handlers: Running operations on change**

[https://docs.ansible.com/ansible/latest/playbook\_guide/playbooks\_handlers.html#handlers](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_handlers.html" \l "handlers)

Refer 21\_handlers.yaml playbook to understand how handlers works in ansible playbook.

**Ansible Roles**

Roles are used to simplify the ansible playbooks. Roles let you automatically load related vars, files, tasks, handlers, and other Ansible artifacts based on a known file structure. After you group your content in roles, you can easily reuse them and share them with other users.

Before we understand the ansible roles, lets first re iterate what are the things ansible playbook content.

Playbook Content:

* Global Declaration
* Variables
* Tasks
* Files
* Templates
* Handlers

**Ansible Role Directory Structure**

