**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**

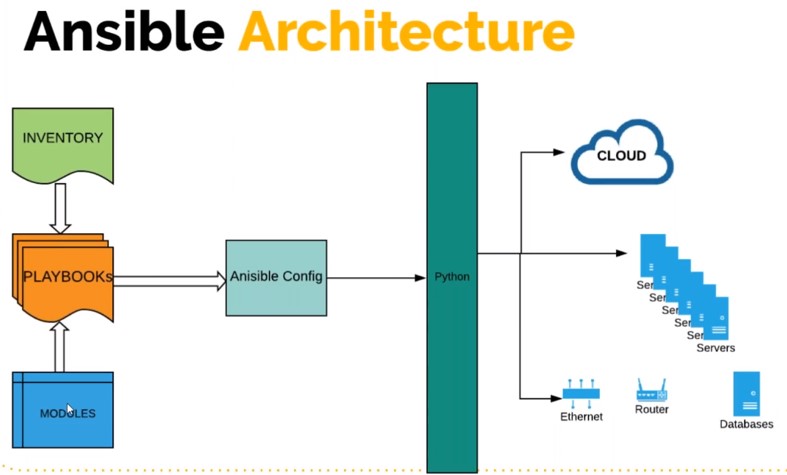
* **Simple:** Ansible is very simple tool.
* **Free:** Ansible is an open-source tool.
* **Very simple to set up and use:** No special [coding](https://www.simplilearn.com/tutorials/programming-tutorial/coding-for-beginners) skills are necessary to use Ansible’s playbooks
* **Powerful:** Ansible lets 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**

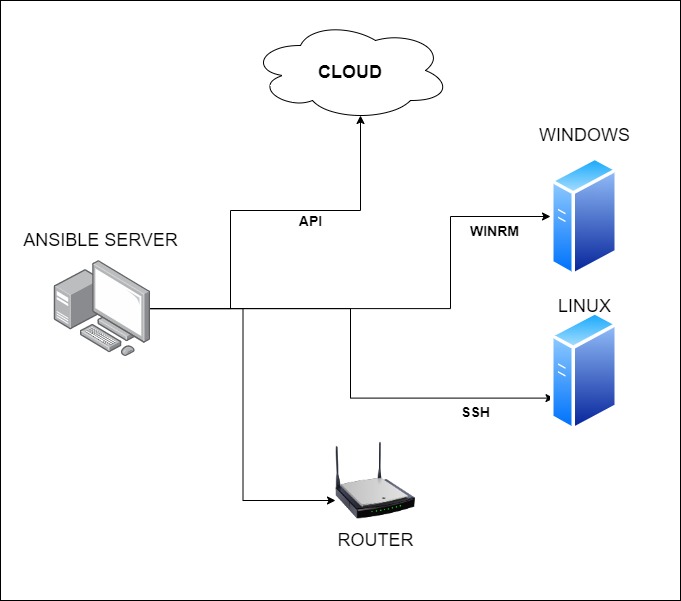
Inventory file defines the hosts and groups of hosts upon which commands, modules, and tasks in a playbook operate. The file can be in one of many formats depending on your Ansible environment and plugins. The default inventory located at **/etc/ansible/hosts**

### **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:**

****

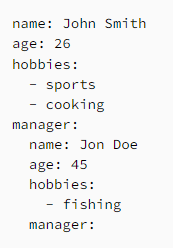
**Control machine**: A control machine is the central node in an Ansible infrastructure. It is used to manage all the other machines in the network.

**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:**

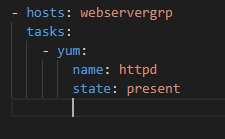
$ ansible -i inventory -m service -a “name=httpd state=started enabled=yes” web01 –become

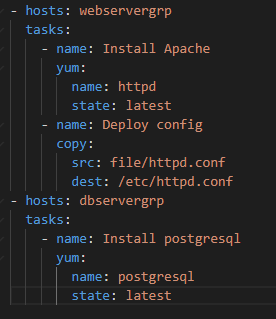
**Example 3:**

$ 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