**Ansible (software)**

-->Ansible is an [**open-source**](https://en.wikipedia.org/wiki/Open-source_software)**automation engine** that automates **software provisioning, configuration management,** and **application deployment**.

-->Ansible has two types of servers: **controlling machines** and **nodes**.  Nodes are managed by a controlling machine over **SSH**.  The controlling machine describes the location of nodes through its [**inventory**](https://en.wikipedia.org/wiki/Ansible_(software)#Inventory_configuration).

-->It does not consume resources because **no daemons or programs are executing for Ansible in the background**, Ansible uses an **agentless** architecture.

**-->**

**Original author** Michael DeHaan

**Developer** Ansible Community / Ansible Inc. / Red Hat Inc.

**Written in** Python, PowerShell

**Operating system** Linux, Unix-like, Windows

**Type** Configuration management, Infrastructure as Code, Orchestration engine

**License** GNU General Public License

**Feature of Ansible:**

**-->**Cross platform support – can support Linux, Unix-like, Windows and network devices etc.

**-->**Agentless architecture, so no work for system admins to manage the agents.

**-->**Written in python

**-->**Use openssh for Unix & Linux systems and WinRM (Windows Remote Management) for communication with windows machines

**-->**Open source and actively maintained by GitHub.

**-->**Very easy installation and configuration.

**-->**Highly scalable.

**-->**Secure as it used ssh for the communication.

**-->**Easy to understand and remember the command syntax. (No coding skill needed)

**Ansible is used for,**

**-->**Configuration Management

**-->**Application deployment

**-->**Provisioning (AWS, Virtual (VMWare, Xen, KVM), Openstack etc)

**User Interfaces:**

**Ansible Tower:**

* Ansible Tower is an API, web service, and web-based console designed to make Ansible usable for IT teams. It is a hub for automation tasks. Tower is a commercial product supported by Red Hat, Inc. One may [register for updates](https://www.ansible.com/open-tower) in releasing Tower as Open Source Software.

**Ansible Tower Pricing:**

**\*Standard**

Up to 100 nodes  
$10,000/year

**\*Premium**

Up to 100 nodes  
$14,000/year

* **AWX Ansible** is an open source community project, sponsored by Red Hat.

**https://www.ansible.com/products/awx-project/faq**

* [**Semaphore**](https://github.com/ansible-semaphore/semaphore) is an open source alternative to Tower, written in [Go](https://en.wikipedia.org/wiki/Go_(programming_language)).

[**https://github.com/ansible-semaphore/semaphore**](https://github.com/ansible-semaphore/semaphore)

**Requirement for Ansible:-**

* SSH Client (OpenSSH)
* Python (PyYAML, Jinja2)

**Lab Setup:-**

Ansible-control ansible.onmo.com

DB Server db.onmo.com

APP server app.onmo.com

**Installation of Ansible: -**

**RPM:** Ansible - available from epel repository

# yum install ansible

**Ansible control account:-**

Username – ansible

Should be created all **client** machines with **sudo** access.

**Password less Connection: -**

# su - Ansible

# ssh-keygen

# for i in 172.19.113.{1..250}; do ssh-copy-id -i /root/.ssh/id\_rsa.pub ansible@$i; done

**Important Files: -**

Inventory file = /etc/ansible/hosts

An example inventory:

[server\_group]

Server1.onmobile.com

Server2.onmobile.com

172.19.113.12

[db\_server]

db.onmo.com

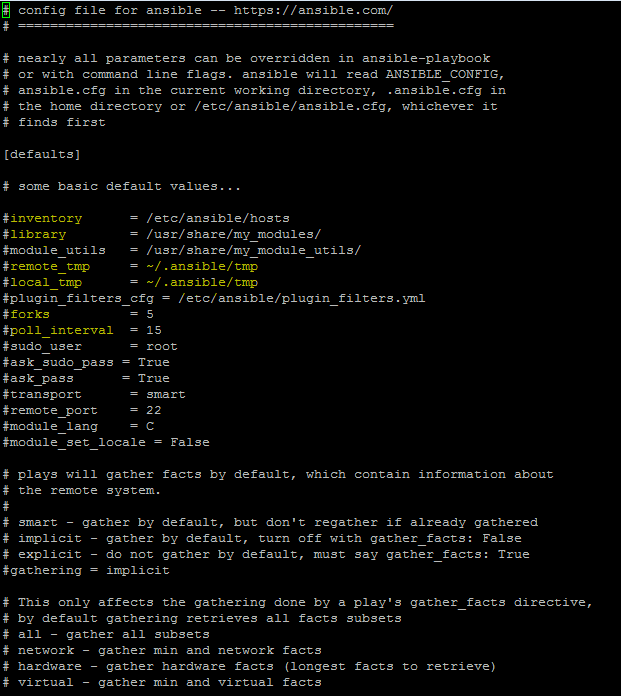
[app\_server]

app.onmo.com

**Configuration file**

# cat /etc/ansible/ansible.cfg

Example:-



**Colours Indications:-**

**Green – Success (change is False) Yellow – Changed (change is True) Red – Failed**

**Modules: -**

Modules are considered to be the units of work in Ansible.

Apart from bare metal hosts, Ansible can deploy to virtualization and cloud environments, including [Amazon Web Services](https://en.wikipedia.org/wiki/Amazon_Web_Services), [Cloud Stack](https://en.wikipedia.org/wiki/CloudStack), Digital Ocean, [Eucalyptus Cloud](https://en.wikipedia.org/wiki/Eucalyptus_(software)), [Google Cloud Platform](https://en.wikipedia.org/wiki/Google_Cloud_Platform), [KVM](https://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine), [Microsoft Azure](https://en.wikipedia.org/wiki/Microsoft_Azure), [OpenStack](https://en.wikipedia.org/wiki/OpenStack), [Rackspace](https://en.wikipedia.org/wiki/Rackspace), [SoftLayer](https://en.wikipedia.org/wiki/SoftLayer), [VMware](https://en.wikipedia.org/wiki/VMware), and [XenServer](https://en.wikipedia.org/wiki/XenServer).

**Commands to list available: -**

ansible-doc -l 🡪 to list all available modules in this release

ansible-doc -s yum 🡪 checking particular Module like a README

**Most Used Syntax Options:**

-m = Module name

-a = argument

--ask-pass = ask for SSH password

--ask-su-pass = ask for su password

--become-method=BECOME\_METHOD privilege escalation method to use (default=sudo),

--become-user=BECOME\_USER run operations as this user (default=None)

-i = INVENTORY

-l = Limit the SUBSET

-o = Limiting the output into one line

-s = run operations with sudo (nopasswd) (deprecated, use Become)

-v = verbose mode

**Basic Administration Commands:**

# ansible server\_group -m ping

# ansible all -m ping

# ansible all -m shell -a "hostname;uname -a;free -m;df -Th"

# ansible all -m shell -a “echo ‘password of server’ | passwd --stdin username” -l servers\_group

# ansible db\_group -m yum -a "name=httpd state=present" -s 🡺 **-s install package with sudo**

# ansible app\_group -m service -a "name=httpd state=started" -s

# ansible all -m copy -a "src=/root/ansibletest dest=/root”

**Playbook:**

Ansible playbooks are a way to send commands to remote computers in a scripted way. Instead of using Ansible commands individually to remotely configure computers from the command line, you can configure entire complex environments by passing a script to one or more systems.

Playbooks express configurations, deployment, and orchestration in Ansible. The Playbook format is [YAML](https://en.wikipedia.org/wiki/YAML). Each Playbook maps a group of hosts to a set of roles. Each role is represented by calls to Ansible tasks.



# cat playbook1.yml

---

- hosts: all

user: ansible

become: yes

become\_method: sudo

tasks:

- name: 1.Install lates version HTTP RPMs

yum: name=httpd state=present

- name: 2.Copy the standard index.html

copy: src=/root/index.html dest=/var/www/html/index.html mode=0664

- name: Add iptables rule if enabled Enabled

command: /sbin/iptables -I INPUT 1 -p tcp --dport http -j ACCEPT -m comment "Http on port 80"

- name: 4. Save iptables rule

command: iptables-save

- name: 5. start httpd service

service: name=httpd state=started enabled=yes

# ansible-playbook /etc/ansible/playbook1.yml **--check** = Checking Play book syntax errors

# ansible-playbook /etc/ansible/playbook1.yml -i /etc/ansible/hosts -l server = running playbook

**Ansible Roles:-**

**What are Ansible playbooks?**

Roles in Ansible are next level abstraction of Ansible playbooks

**Benefits of Ansible**

* Idea of include files and combine them to form clean, reusable abstractions
* Easy to maintain/troubleshooting playbooks

**Structure of Roles:-**

**Files: contains regular files those needs to copy to target hosts.**

**Handlers: event handlers**

**Meta: role dependencies**

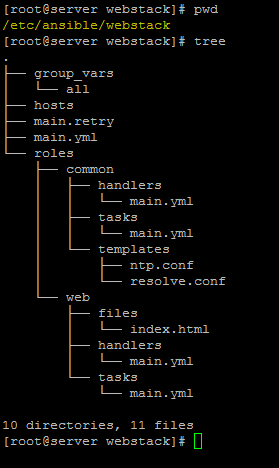
**Templates: similar to files, but it contains dynamic data**

**Tasks: playbook tasks**

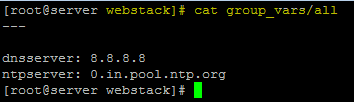
**Vars/group\_vars: variable definitions**

# **ansible-galaxy init webstack**

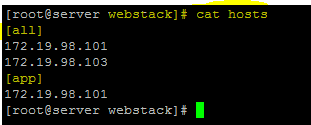
- webstack was created successfully



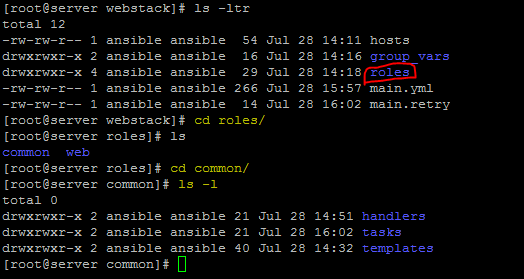
group\_vars 🡪 contains all variables



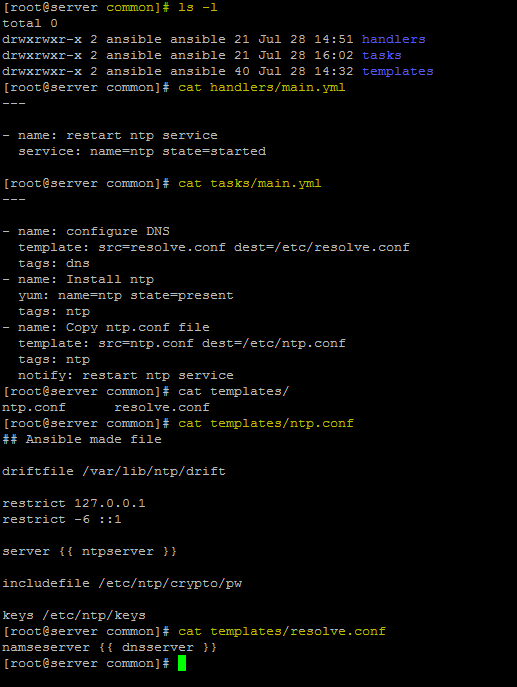
hosts 🡪 contains all host informations

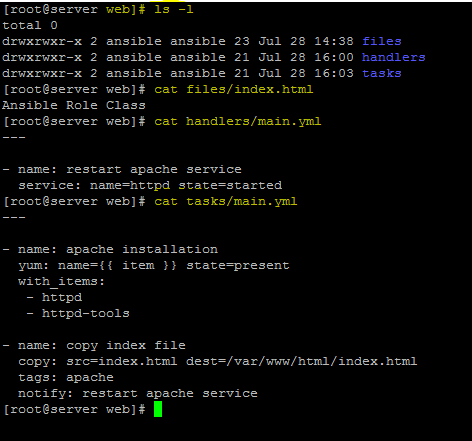


Roles 🡪 contains all tasks, handlers, templates and files



Common role to apply all Host and web role only for application server



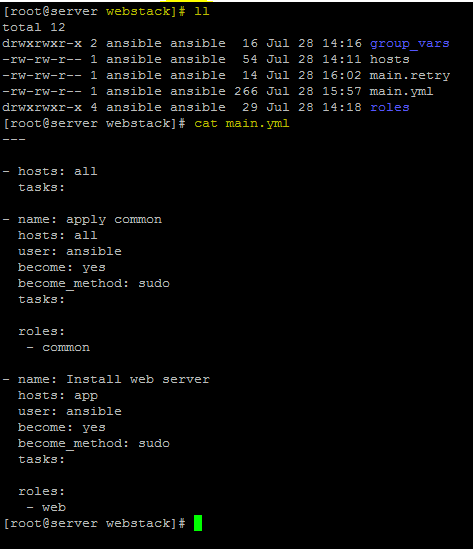


**Notes:**

Item – with list of items like multiple packages can be install

notify should match with handles name

**Mail.yml** have all information about hosts, user and roles.

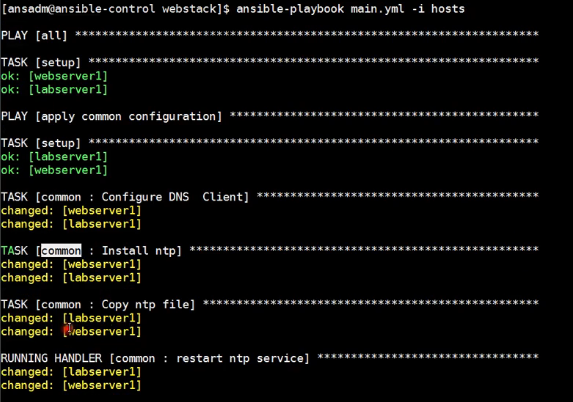


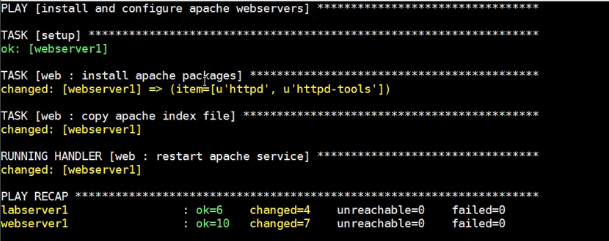
**Run Playbook using below command**

**# ansible-playbook main.yml -i hosts**

**# ansible-playbook main.yml -i hosts -l appserver --tags dns** 🡺 it will do only dns tasks and can limit server using -l options

**Sample Output:**





**Verify in nodes:-**

**All nodes:-**

# cat /etc/resolve.conf

# ntpq -p

**App node:-**

# rpm -qa | grep httpd

# cat /var/www/html/index.html

# systemctl status httpd

**Ansible for OS patch management on CentOS linux**

Checking the server’s which are communicating with my Ansible Master server,

Ex: we have 3 servers here.

**[root@ansible-master ansible]# cat hosts**

[servergroup]

testserver1

testserver2

[root@ansible-master ansible]# ansible servergroup -m shell -a “uname -a;uptime”

[root@ansible-master ansible]# ansible servergroup -m shell -a “ps -eaf | grep apache”

[root@ansible-master ansible]# ansible servergroup -m shell -a “systemctl stop httpd.service”

**[root@ansible-master ansible]# cat Linuxpatch.yml**

—

## Demo Ansible Playbook to perform patching on RHEl/CentOS Server

## For Demo Purpose only.. Use it at your own responsibility.

## Can update it as per your requirements

– hosts: servergroup

become\_user: root

serial: 2

tasks:

# purpose of this task to check if application is running or stopped

– name: verify application/database processes are not running

shell: if ps -eaf | egrep ‘apache|http’|grep -v grep > /dev/null ;then echo ‘process\_running’;else echo ‘process\_not\_running’;fi

ignore\_errors: true

register: app\_process\_check

# this task is decision,play will fail/quit,if application is running

– name: decision point to start patching

fail: msg=”{{ inventory\_hostname }} have running Application.Please stop the application first, then attempt patching.”

when: app\_process\_check.stdout == “process\_running”

# this task will upgrade/install the rpm’s if application is stopped

– name: upgrade all packages on the server

yum:

name=”kernel”

state=latest

when: app\_process\_check.stdout == “process\_not\_running” and ansible\_distribution == ‘CentOS’ or ansible\_distribution == ‘Red Hat Enterprise Linux’

register: yum\_update

# this task is to check if kernel update happend and system needs reboot or not

– name: check if reboot required after kernel update.

shell: KERNEL\_NEW=$(rpm -q –last kernel |head -1 | awk ‘{print $1}’ | sed ‘s/kernel-//’); KERNEL\_NOW=$(uname -r); if [[ $KERNEL\_NEW != $KERNEL\_NOW ]]; then echo “reboot\_needed”; else echo “reboot\_not\_needed”; fi

ignore\_errors: true

register: reboot\_required

# this task is to restart the system

– name: restart system

command: shutdown -r +1 “Rebooting System After Patching”

async: 0

poll: 0

when: reboot\_required.stdout == “reboot\_needed”

register: reboot\_started

ignore\_errors: true

# this task is to wait for 3 minutues for system to come up after the reboot

– name: pause for 180 secs

pause:

minutes: 3

# this task is to confirm,system is up and responding to ssh

– name: check if system responding to ssh

local\_action:

module: wait\_for

host={{ inventory\_hostname }}

port=22

delay=15

timeout=300

state=started

when: reboot\_started|changed

**Run playbook**

**# ansible-playbook Linuxpatch.yml**

**Get Ansible variables using below commands**

**# ansible -m setup all**

**# ansible -m setup all |grep ansible\_distribution**

"ansible\_distribution": "CentOS",

"ansible\_distribution\_file\_parsed": true,

"ansible\_distribution\_file\_path": "/etc/redhat-release",

"ansible\_distribution\_file\_variety": "RedHat",

"ansible\_distribution\_major\_version": "7",

"ansible\_distribution\_release": "Core",

"ansible\_distribution\_version": "7.2.1511",

"ansible\_distribution": "CentOS",

"ansible\_distribution\_file\_parsed": true,

"ansible\_distribution\_file\_path": "/etc/redhat-release",

"ansible\_distribution\_file\_variety": "RedHat",

"ansible\_distribution\_major\_version": "6",

"ansible\_distribution\_release": "Final",

"ansible\_distribution\_version": "6.9",

"ansible\_distribution": "CentOS",

"ansible\_distribution\_file\_parsed": true,

"ansible\_distribution\_file\_path": "/etc/redhat-release",

"ansible\_distribution\_file\_variety": "RedHat",

"ansible\_distribution\_major\_version": "5",

"ansible\_distribution\_release": "Final",

"ansible\_distribution\_version": "5.3",

"ansible\_distribution": "CentOS",

"ansible\_distribution\_file\_parsed": true,

"ansible\_distribution\_file\_path": "/etc/redhat-release",

"ansible\_distribution\_file\_variety": "RedHat",

"ansible\_distribution\_major\_version": "5",

"ansible\_distribution\_release": "Final",

"ansible\_distribution\_version": "5.3",

**Ansible Automation | Apply CIS Security BenchMarking for RHEL/ CentOS**

Find below role and change it accordingly

[**https://github.com/MindPointGroup/RHEL7-CIS**](https://github.com/MindPointGroup/RHEL7-CIS)

[**https://github.com/major/cis-rhel-ansible**](https://github.com/major/cis-rhel-ansible)

[**https://github.com/HarryHarcourt/Ansible-RHEL7-CIS-Benchmarks**](https://github.com/HarryHarcourt/Ansible-RHEL7-CIS-Benchmarks)

**Ansible Tips and Tricks**

Check syntax of the playbook

**# Ansible-playbook playbook.yml - -syntax-check**

Execute playbook in the check (dry-run) mode, which check what changes will be performed.

**# Ansible-playbook playbook.yml - -check**

List hosts on which playbook will be executed

**# Ansible-playbook playbook.yml - -list-hosts -l subset**

List tags in the playbook

**# Ansible-playbook playbook.yml - -list-tags**

Only run plays and tasks tagged with these tag values

**# Ansible-playbook playbook.yml - -tags tag1,tag2…tagN**

Skip the tasks associated with specific tasks

**# Ansible-playbook playbook.yml - -skip-tags tag1,tag2,…tagN**

The - -forks what lets Ansible run on multiple hosts in parallel, NUM is specified as an integer, the default is 5.

**# Ansible-playbook playbook.yml - -forks=NUM**

Run a playbook on the target hosts without inventory files.

**# Ansible-playbook playbook.yml -i [IP/Hostname],[IP/Hostname],**

**And Ref below,**

[**https://blog.ippon.tech/ansible-tips-and-tricks/**](https://blog.ippon.tech/ansible-tips-and-tricks/)