**What is Ansible?**

Ansible is an IT automation tool. It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates.

Ansible’s main goals are simplicity and ease-of-use. It also has a strong focus on security and reliability, featuring a minimum of moving parts, usage of OpenSSH for transport (with other transports and pull modes as alternatives), and a language that is designed around auditability by humans–even those not familiar with the program.

Ansible manages machines in an agent-less manner. There is never a question of how to upgrade remote daemons or the problem of not being able to manage systems because daemons are uninstalled. Because OpenSSH is one of the most peer-reviewed open source components, security exposure is greatly reduced. Ansible is decentralized–it relies on your existing OS credentials to control access to remote machines. If needed, Ansible can easily connect with Kerberos, LDAP, and other centralized authentication management systems.

[**Prerequisites**](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#id9)

[**Control node requirements**](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#id10)

**SSH ,Python 2 or Python 3**

When choosing a control node, bear in mind that any management system benefits from being run near the machines being managed. If you are running Ansible in a cloud, consider running it from a machine inside that cloud. In most cases this will work better than on the open Internet.

[**Managed node requirements**](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#id11)

By default this uses SFTP. If that’s not available, you can switch to SCP in [ansible.cfg](https://docs.ansible.com/ansible/latest/reference_appendices/config.html" \l "ansible-configuration-settings)

Disable selinux

By default, Ansible uses the Python interpreter located at /usr/bin/python to run its modules. However, some Linux distributions may only have a Python 3 interpreter installed to /usr/bin/python3 by default. On those systems, you may see an error like:

"module\_stdout": "/bin/sh: /usr/bin/python: No such file or directory\r\n"

Ansible’s [raw module](https://docs.ansible.com/ansible/latest/modules/raw_module.html#raw-module), and the [script module](https://docs.ansible.com/ansible/latest/modules/script_module.html#script-module), do not depend on a client side install of Python to run. Technically, you can use Ansible to install a compatible version of Python using the [raw module](https://docs.ansible.com/ansible/latest/modules/raw_module.html#raw-module), which then allows you to use everything else. For example, if you need to bootstrap Python 2 onto a RHEL-based system, you can install it as follows:

$ ansible myhost --become -m raw -a "yum install -y python2"

Installing Ansible

sudo yum install epel-release

[Getting the latest configuration](https://docs.ansible.com/ansible/latest/installation_guide/intro_configuration.html#id5)

If installing Ansible from a package manager, the latest ansible.cfg file should be present in /etc/ansible, possibly as a .rpmnew file (or other) as appropriate in the case of updates.

If you installed Ansible from pip or from source, you may want to create this file in order to override default settings in Ansible.

cd /etc/ansible/

/usr/share/ansible/plugins

ansible –version

Components of Ansible::

Programs

Ansible

Ansible-doc

Ansible-playbook

Ansible-pull

Modules

Perform configuration and system management like Modules copy, service, file etc.

Set of instructions written for particular service type.

Ansible is available in EPEL repository.

sudo yum -y update

sudo yum install epel-release

sudo yum -y install ansible

To check repos yum repolist

rpm -qa | grep ansible

rpm -ql ansible-2.9.7-1.el7.noarch | more

rpm -ql ansible-2.9.10-1.el7.noarch | more

/etc/ansible

/etc/ansible/ansible.cfg

/etc/ansible/hosts

/etc/ansible/roles

/usr/bin/ansible

/usr/bin/ansible-2

/usr/bin/ansible-2.7

/usr/bin/ansible-config

/usr/bin/ansible-connection

/usr/bin/ansible-console

/usr/bin/ansible-console-2

/usr/bin/ansible-console-2.7

/usr/bin/ansible-doc

/usr/bin/ansible-doc-2

/usr/bin/ansible-doc-2.7

/usr/bin/ansible-galaxy

/usr/bin/ansible-galaxy-2

/usr/bin/ansible-galaxy-2.7

/usr/bin/ansible-inventory

/usr/bin/ansible-playbook

/usr/bin/ansible-playbook-2

/usr/bin/ansible-playbook-2.7

/usr/bin/ansible-pull

/usr/bin/ansible-pull-2

/usr/bin/ansible-pull-2.7

/usr/bin/ansible-test

/usr/bin/ansible-vault

/usr/bin/ansible-vault-2

/usr/bin/ansible-vault-2.7

/usr/lib/python2.7/site-packages/ansible

/usr/lib/python2.7/site-packages/ansible-2.9.10-py2.7.egg-info

/usr/lib/python2.7/site-packages/ansible-2.9.10-py2.7.egg-info/PKG-INFO

/usr/lib/python2.7/site-packages/ansible-2.9.10-py2.7.egg-info/SOURCES.txt

/usr/lib/python2.7/site-packages/ansible-2.9.10-py2.7.egg-info/dependency\_links.txt

/usr/lib/python2.7/site-packages/ansible-2.9.10-py2.7.egg-info/not-zip-safe

/usr/lib/python2.7/site-packages/ansible-2.9.10-py2.7.egg-info/requires.txt

/usr/lib/python2.7/site-packages/ansible-2.9.10-py2.7.egg-info/top\_level.txt

/usr/lib/python2.7/site-packages/ansible/\_\_init\_\_.py

/usr/lib/python2.7/site-packages/ansible/\_\_init\_\_.pyc

/usr/lib/python2.7/site-packages/ansible/\_\_init\_\_.pyo

/usr/lib/python2.7/site-packages/ansible/cli

/usr/lib/python2.7/site-packages/ansible/cli/\_\_init\_\_.py

/usr/lib/python2.7/site-packages/ansible/cli/\_\_init\_\_.pyc

/usr/lib/python2.7/site-packages/ansible/cli/\_\_init\_\_.pyo

/usr/lib/python2.7/site-packages/ansible/cli/adhoc.py

/usr/lib/python2.7/site-packages/ansible/cli/adhoc.pyc

/usr/lib/python2.7/site-packages/ansible/cli/adhoc.pyo

/usr/lib/python2.7/site-packages/ansible/cli/arguments

/usr/lib/python2.7/site-packages/ansible/cli/arguments/\_\_init\_\_.py

/usr/lib/python2.7/site-packages/ansible/cli/arguments/\_\_init\_\_.pyc

/usr/lib/python2.7/site-packages/ansible/cli/arguments/\_\_init\_\_.pyo

/usr/lib/python2.7/site-packages/ansible/cli/arguments/option\_helpers.py

/usr/lib/python2.7/site-packages/ansible/cli/arguments/option\_helpers.pyc

/usr/lib/python2.7/site-packages/ansible/cli/arguments/option\_helpers.pyo

/usr/lib/python2.7/site-packages/ansible/cli/config.py

/usr/lib/python2.7/site-packages/ansible/cli/config.pyc

/usr/lib/python2.7/site-packages/ansible/cli/config.pyo

Create Ansible user

useradd -d /home/ansadm ansadm -m ansadm

SET Password to non expiry

passwd -x -1 ansadm

create admin password on all servers

Now Set up ssh keys and enable passwrdless login into your cluster

Ssh-keygen -t rsa

cat id\_rsa.pub

chmod 600 authorized\_keys

id ansadm

There is no need to install ansible on any clients.

## Important terms used in Ansible

### Ansible server:

The machine where Ansible is installed and from which all tasks and playbooks will run.

### Machine:

### A physical machine, a VM or a container

### Target Machine:

### A machine where things need to be installed by Ansible

### TASK:

### An action to be done

### Playbook:

### A location where YAML files are written and executed

### Ansible location of various directories:

ansible --version

ansible 2.9.7

config file = /etc/ansible/ansible.cfg

configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']

ansible python module location = /usr/lib/python2.7/site-packages/ansible

executable location = /bin/ansible

python version = 2.7.5 (default, Apr 2 2020, 13:16:51) [GCC 4.8.5 20150623 (Red Hat 4.8.5-39)]

PATH for module libraries in python

/usr/lib/python2.7/site-packages/ansible/modules

Path for ansible command

/usr/bin/

Lets set up our servers for ssh password less communication

First install openssh

Allow permit root login yes and allow authentication with password yes

Generate key on all servers

ssh-keygen -t rsa

Now copy the public key to your local machine authrozied key

cat .ssh/id\_rsa.pub >> .ssh/authorized\_keys

ssh-copy-id root@ec2-3-82-153-112.compute-1.amazonaws.com

sudo vi /etc/ssh/sshd\_config

sudo systemctl restart sshd

YAML:

Yet another Mark-up language. Readable programming language.

Used to express Ansible playbooks:

It has key value pairs

Dictionary is represented in key value pair.

Example :

James:

Name: James Hunter

rollNo: 23

div: B

sex : Male

Lists:

Each element is in list start with –

Countries:

-India

-china

-peru

List Inside Dictionary:

Example:

James :

Name: James Hunter

rollNo: 23

div: B

sex : Male

Countries:

-India

-china

-peru

Anisble configuration and adhoc commands

Ansible inventory file ::

/etc/ansible/hosts

[hadoopCluster]

172.31.17.153

172.31.27.111

172.31.2.134

172.31.53.193

172.31.30.99

172.31.5.136

Configuration file is :: /etc/ansible/ansible.cfg

Anisble mainly works on modules a

Which can be written in python or yaml

Ansible-doc -l

ansible-doc -l | wc -l

3387

TO see what functions can a particular module do ::

ansible-doc -s yum

Adhoc commands

ansible hadoopCluster -m ping

ansible all -m ping

ansible all -m ping -o

ansible hadoopluster -m shell -a "uname -a;df -h" -o

ansible hadoopluster -m shell -a "uname -a;df -h" -o -v

install a package

ansible hadoopCluster -m yum -a “name=httpd state=present” -s

ansible hadoopCluster -m service -a "name=httpd state=started” -s

ansible hadoopCluster -m service -a "name=nmap state=present” -s

ansible hadoopCluster -m copy -a "src=/../ dest=/tmp/” -s

Ansible playbook

**Ansible playbooks**

Files where anisble code is written.Playbooks are written in YAML format.YAML stands for YET ANOTHER MAKRUP LANGAUGE.IT is one of core features of Ansible and tell Anisble what to execute.

They are to do lists for ansible and they contains a list of tasks.

Playbook contains steps a user want to execute on a particular machine.

They run sequentially.They are building blocks for all the use cases of all Ansible.

Playbook Structure ::

Each playbook contains aggregation of one or more plays. They are structured using plays and there can be more than one play inside a playbook.

YAML is a strict type cast language.so extra care should be taken while writing a playbook.

A yAML start with ---(three hypens)

---

name: install and configure DB

hosts: testServer

become: yes

vars:

oracle\_db\_port\_value : 1521

tasks:

-name: Install the Oracle DB

yum: <code to install the DB>

-name: Ensure the installed service is enabled and running

service:

name: <your service name>

The different YAML Tags ::

NAME:: This tag tells what actually what the playbook is going to do.

Hosts:: hosts or group of hosts where we want to run the command.

Vars:: vairables you want to run in playbook ,similar to variable in any programming language.

TASKS:: As name suggest all playbooks contain them and they are executed to perform certain actions.

 A tasks field contains the name of the task. This works as the help text for the user. It is not mandatory but proves useful in debugging the playbook. Each task internally links to a piece of code called a module. A module that should be executed, and arguments that are required for the module you want to execute

**Ansible Roles: Next level abstraction of ansible playbook.**

Benefits of ansible roles:

Idea of include file and combine them to form clean and reusable abstraction.

Easy to maintain /troubleshoot the playbooks.

Structure of roles:

Files: contains regular files those need to be copied on the target hosts.

Handlers:: event handlers

Meta: role dependencies

Template :: similar to file but contains dynamic data -you can point to variables

Tasks : playbook tasks

Var/groups : variable definitions

cd /etc/ansible/

[root@ip-172-31-29-190 ansible]# tree

.

├── ansible.cfg

├── hosts

└── roles

Ansible roles consists of many playbooks, which is similar to modules in puppet and cook books in chef. We term same in ansible as roles.

Roles are a way to group multiple tasks together into one container to do automation in very effective manner with clean directory structure.

Role are group of tasks and files for a certain role to break down the configuration.

Role can be easily used if it is suitable to someone else.

Can be easily modified and will reduce syntax error.

Creating ansible error:

Use ansible galaxy error to create apache role

ansible-galaxy init /etc/ansible/roles/apache –offline

tree /etc/ansible/roles/apache/

/etc/ansible/roles/apache/

├── defaults

│   └── main.yml

├── files

├── handlers

│   └── main.yml

├── meta

│   └── main.yml

├── README.md

├── tasks

│   └── main.yml

├── templates

├── tests

│   ├── inventory

│   └── test.yml

└── vars

└── main.yml

Directory Structure ::

We have got clear directory structure where each directory contains main.yml which contains main content.

1. Tasks: contains main list of tasks to be executed by the role.
2. Handlers: contains handlers, which may be used by this role or even anywhere outside this role.
3. Defaults: default variables for the role.
4. Vars: other variables for the role ,vars have higher priority then defaults.
5. Files: contains files required to transfer or deployed to the target machines via a role.
6. Templates: contains templates which can be deployed via a role.
7. Meta: defines some data and information about role. —author, dependency, versions, examples.
8. Go to tasks
9. Eg

[root@learnitguide apache]# vi tasks/main.yml  
---  
- name: Install httpd Package  
  yum: name=httpd state=latest  
- name: Copy httpd configuration file  
  copy: src=/data/httpd.original dest=/etc/httpd/conf/httpd.conf  
- name: Copy index.html file  
  copy: src=/data/index.html dest=/var/www/html  
  notify:  
  - restart apache  
- name: Start and Enable httpd service  
  service: name=httpd state=restarted enabled=yes

We can also use import\_tasks and break the configuration

[root@learnitguide apache]# cat tasks/main.yml  
---  
# tasks file for /etc/ansible/roles/apache  
- import\_tasks: install.yml  
- import\_tasks: configure.yml  
- import\_tasks: service.yml

Files::

Copy the required files into file directory.

[root@learnitguide apache]# ll files/\*  
-rw-r--r-- 1 root root 11753 Feb  4 10:01 files/httpd.conf  
-rw-r--r-- 1 root root    66 Feb  4 10:02 files/index.html  
[root@learnitguide apache]# cat files/index.html  
This is a homepage created by learnitguide.net for ansible roles.  
[root@learnitguide apache]#

Handlers : as we have already defined in task to notify if something changes . Use same name and put it in main.yml file.

[root@learnitguide apache]# cat handlers/main.yml  
---  
# handlers file for /etc/ansible/roles/apache  
- name: restart apache  
  service: name=httpd state=restarted

Meta :: add below information in main.yml of meta folder:

[root@learnitguide apache]# cat meta/main.yml  
galaxy\_info:  
  author: LearnItGuide.net  
  description: Apache Webserver Role  
  company: LearnITGuide.net  
  # If the issue tracker for your role is not on github, uncomment the  
  # next line and provide a value  
  # issue\_tracker\_url: http://example.com/issue/tracker  
  # Some suggested licenses:  
  # - BSD (default)  
  # - MIT  
  # - GPLv2  
  # - GPLv3  
  # - Apache  
  # - CC-BY  
  license: license (GPLv2, CC-BY, etc)  
  min\_ansible\_version: 1.2  
  # If this a Container Enabled role, provide the minimum Ansible Container version.  
------skipped

Now after creating all files and directories we will create another playbook to excute the role

root@learnitguide apache]# cat /etc/ansible/runsetup.yml  
---  
 - hosts: node2  
   roles:  
   - apache  
[root@learnitguide apache]#

also verify errors

[root@learnitguide apache]# ansible-playbook /etc/ansible/runsetup.yml --syntax-check  
playbook: /etc/ansible/runsetup.yml  
[root@learnitguide apache]#

ansible-playbook /etc/ansible/runsetup.yml

Check the service

**Create ec2 server using ansible**

Creates or terminates ec2 instances.

For working on Ansible we need to first set up a few things,

* AWS user account
* Ansible
* Python
* Boto

**Boto:**(Boto is a Python package which provides an interface to AWS.)

1. First, install pip
   1. *$ sudo apt install python3-pip or*
   2. *$ yum install python-pip*
2. Now install boto
   1. *$ pip install boto*

We can install others with documentation already.

Now open a terminal and create a file with the extension .yml

# Basic provisioning example

- name: Ansible test

hosts: localhost

tasks:

- name: launching AWS instance using Ansible

ec2:

key\_name: aws\_instance\_Ansible

instance\_type: t2.micro

image: ami-0dacb0c129b49f529

region: us-east-2

wait: yes

group: Ansible

count: 1

vpc\_subnet\_id: default

assign\_public\_ip: yes

aws\_access\_key: \*\*\*\*\*\*\*\*\*\*\*xxxxxxxx

Aws\_secret\_key: \*\*\*\*\*\*\*\*\*\*\*xxxxxxxx

**Hosts:** add [webserver] localhost in /etc/Ansible/hosts file as my internet is running on the local server. If the file does not exist create one at the same location then add

**Key\_name:** Go to EC2 dashboard -> Key pairs -> Create key pair -> Copy key pair name

**Instance\_type:** You can select the instance type whichever you want to launch. Go to EC2 dashboard -> Launch instance -> Check instance type.

**Image:** Go to EC2 dashboard -> Launch instance -> ami id (Image id)

**Vpc\_subnet\_id:** I made it default as I don’t any VPC configuration.

Add your **aws\_access\_key** and **aws\_secret\_key** which you got from IAM user creation. The rest are the basic details. If you want more details you can visit the [Ansible official website](https://docs.ansible.com/ansible/latest/modules/ec2_module.html).

Run below command to check whether Ansible is ready to launch EC2 or not.

Ansible-playbook -C filename.yml

Where -C will check if everything is ready or not.

Once everything looks good, run below command and within a minute your EC2 server will be launched.

Ansible-playbook filename.yml

User management through Ansible:

Though user management is a day to day task for system admin which is among the simpler task but in a big data cluster it become cumbersome .

As sometimes you donot have central user management tool such as AD or LDAP.

How to create local users for system/application when AD is not integrated.

With Ansible you need to do is create few tasks in ansible playbook and a user is created in few seconds.

Our playbook example contains ::

Create a new user

Create local groups

Create ssh keys for user.

Update pub keys in authorised keys file

Delete local group

Delete local users

Sample playbook for same:

---

## this is a playbook for user management using Ansible

## playbook uses “user,group,authrorized keys files”

-hosts : ALL

Become user:root

tasks:

#this task creates new group

* + Name :add a group

Group:

name={{items}}

state=present

with\_items:

* + Demogrp
  + Demogrp1
  + Demogrp2

Tags: add new groups

#this tasks add users to several target machines

* + Name: add several users

User:

Name={{item}}

Groups=demogrp1

Password =””

Shell=/bin/bash

With items:

* + Demouser1
  + Demouser2
  + Demouser3

Tags : add new users

#this tag is used to generate ssh keys

-name :generate ssh keys

user:

name = {{ item }}

generate\_ssh\_keys = yes

ssh\_key\_bites=4096

ssh\_key\_file = .ssh/id\_rsa

with\_items:

-demouser1

Tags : generate\_ssh\_keys

## this task is to copy public keys to target servers

-name: update authrised keys for a user

Authorized\_keys:

User:demouser1

State: present

Key : “{{ lookup(‘file’ , ‘/home/demouser1/.ssh/id\_rsa.pub) }}”

Tags : copy\_pub\_keys

##this task delete the users

* + Name: delete several users

User:

Name:{{ item}}

State=absent

With\_items:

-demouser3

-demouser4

Tags : remove\_users

##this tasks is to delete the group

-name: delete several group

Group:

Name = {{ item }}

State = absent

With\_items:

* + Demogrp1
  + Demogrp2

Tags: remove group

Ansible-playbook usermanagement.yaml --list-tags

Ansible-playbook usermanagement.yaml --tags add\_new\_group

Cat /etc/group

Ansible-playbook usermanagement.yaml --tags add\_new\_user

Ansible-playbook usermanagement.yaml --tags generate\_ssh\_keys

Userdel -r demouser1

Ansible-playbook usermanagement.yaml --tags copy\_pub\_key

Ansible-playbook usermanagement.yaml --tags remove\_user,remove\_group

**Ansible operating system patch management on Linux :**

---

##Anisble playbook to perform patching on RHEL/CENTOS server

##for demo purpose only.

- hosts:[NonHadoop]

become\_users:root

serial: 2

tasks:

#purpose of this taks is to check if applications is running or stopped

- name: verify applications/databases processes are not running

shell: if ps -eaf | egrep 'apacche|httpd'|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 if application is topped

- 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 decide if the system is updated and we need to reboot the system or not

- name: check if reboot is need or not

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 minutes for system to come up after reboot

- name: pause for 180 seconds

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

when: reboot\_started|changed

ansible -m setup all | grep ansible\_distribution

The authenticity of host '172.31.29.190 (172.31.29.190)' can't be established.

ECDSA key fingerprint is SHA256:DsaggT9R7CwjAxIusedo9K3a98sLysi7FGQXw/Gn06I.

ECDSA key fingerprint is MD5:ec:e8:b6:0c:f7:d2:b1:42:f1:27:32:38:2b:d0:25:6d.

Are you sure you want to continue connecting (yes/no)? yes

"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": "7.8",

"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": "7.8",

[root@ip-172-31-29-190 ansible]# ansible -m setup NonHadoop | 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": "7.8",

"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": "7.8",

ansible -m ping all

ansible NonHadoop -m shell -a "uname -a;uptime"

172.31.53.193 | CHANGED | rc=0 >>

Linux ip-172-31-53-193.ec2.internal 3.10.0-1127.el7.x86\_64 #1 SMP Tue Mar 31 23:36:51 UTC 2020 x86\_64 x86\_64 x86\_64 GNU/Linux

17:53:33 up 3:04, 1 user, load average: 0.00, 0.01, 0.05

172.31.29.190 | CHANGED | rc=0 >>

Linux ip-172-31-29-190.ec2.internal 3.10.0-1127.13.1.el7.x86\_64 #1 SMP Tue Jun 23 15:46:38 UTC 2020 x86\_64 x86\_64 x86\_64 GNU/Linux

17:53:33 up 3:05, 2 users, load average: 0.05, 0.05, 0.04

# **Ansible Automation | Perform Password Hashing and use it with Ansible**

Password Hashing ::

Is a process of taking variable length password as input and creating a cryptic , fixed length password from it suing cryptic mechanism.

To create hash password we use salt, salt is generally a random value used to generate hashed password.

Benefits of Hashing:

More security with hashed password

Hashing is one-way function

Hashed password can be used to manage users through automation tools.

Methods to generate Hashed Passwords:

Different alogriths :: MD5,SHA and sha-512

Using python :

Python -c ‘import crypt,getpass; print.crypt.crypt(getpass.getpass())

This command with add random salt and prompt used to type password.uses sha-552

Using openssl

Openssl passwd -1 -salt $(openssl rand -base64 6) SecurePass

Sample hash playbook using hash password ::

---

-hosts : all

user: root

become: yes

become\_method:sudo

vars:

user\_pass: HashedUserPass

root\_pass: HasedRootPass

tasks:

- name: create new user

user:

name:SK

uid: 1112

comment: SK

password: "{{ user\_pass }}"

shell: /bin/bash

tags: create\_user

- name: root password changes

user: name=root.update\_password=always password="{{ root\_pass }}"

tags: update\_root\_pass

~

~

Ansible-playbook hashinh.yml --list-tags

Ansible-playbook hashinh.yml -t create\_user

Change -l Sk

Cat /etc/shadow

# **Ansible Automation | NTP Client Configuration and Management using Ansible**

Configuring and managing the NTP client on enterprise servers is one of the tasks of system administrators.

NTP client is responsible for the time sync of client server with time sources. Doing it manually can take time and cause problem.

Below mentioned is a small playbook to install manage and configure the NTP client on linux servers . Modules used are:

YUM

TEMPLATE

SERVICE

We will have variables and handlers to make this playbook more efficient.

ansible -i hosts -m shell -a "ntpq -p" all

---

## this is demo playbook for NTP client config using Ansible.

## playbook use yum,service and template modelues

- hosts: NonHadoop

become\_user:roots

vars:

ntp\_server1:ntp1.syrahost.com

ntp\_server2:ntp2.syrahost.com

tasks:

# checck if crony is installed ,if yes rmoeve it.

- name:remove crony client package from the system

yum:

name=chrony

state=absent

tags: remove\_chrony,start\_ntp

# tasks is to ensure ntp is installed

- name: checck/install ntp client package

yum:

name=ntp

state=present

tags: install\_ntp,start\_ntp

# this is to copy the custom ntp.conf file

- name: deploy ntp.conf file to the clients

template:

src=template/ntp.j2

dest=/etc/ntp.conf

owner=root

group=root

mode=0644

backup=yes

notify: start\_restart\_ntp\_client

tags: copy\_ntp\_conf,start\_ntp

handlers:

# make sure ntp service is restarted

- name: start\_restart\_ntp\_client

service:

name=httpd

state=restarted

enabled=yes

ansible-playbook ntpplaybook.yml

# **Ansible Automation | Apply CIS Security BenchMarking for RHEL/ CentOS 6**

**CIS Benchmarks** are best practices for the secure configuration of a target system. Available for 100+ **CIS Benchmarks** covering more than 14 technology groups, **CIS Benchmarks** are developed through a unique consensus-based process comprised of cybersecurity professionals and subject matter experts around the world.

<https://www.youtube.com/watch?v=huR0UvaTXQM&list=PLLsor6GJ_BEEC9jUSc760iqaOx6u5lqRA&index=11>

comma separated tags for differet tags

As a system/build engineer we spend lot of time on searching and applying the security recommendations for RHEL/CentOS SOE images. In this video demo is on Ansible CIS benchmark role written by Major Hayden

Ansible tricks ::

## list all tasks in the playbook

Ansible-playbook playbook.yml –list-tasks

## start the playbook from a particular task

Ansible-playbook playbook.yml --start-at-task=”task name”

## start the playbook step by step from interactive way.This will prompt the user for to confirm each task before running

Ansible-playbook playbook.yml –step

##check syntax of playbook

Ansible-playbook playbook.yml --syntax-check

## execute the playbook in the check –dry run mode, which checks what changes are going to be performed.

Ansible-playbook playbook.yml --check

## list tags of a playbook

Ansible-playbook playbook.yml –list-tags

## list hosts

Ansible-playbook playbook.yml --list-hosts -1 subset

## only run tasks and plays tagged in these tag values

Ansible-playbook playbook.yml --tags tag1,tag2,tag3,….tagN

##skip tasks associated with specific tags

Ansible-playbook playbook.yml --skip-tags tag1,tag2,tag3…tagN

##the fork lets ansible to run playbook 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 | ServerNAme]

Ansible -m ping all