How to Install and Configure ‘Ansible’ Automation Tool for IT Management

#### Step :1 Set EPEL repository

Ansible package is not available in the default yum repositories, so we will enable epel repository for CentOS 7 using below commands

[root@ansible ~]# rpm -iUvh http://dl.fedoraproject.org/pub/epel/7/x86\_64/e/epel-release-7-8.noarch.rpm

#### Step:2 Install Anisble using yum command

[root@ansible ~]# ansible --version

#ansible --version

#### Step:3 Setup keys based SSH authentication with Nodes.

root@ansible ~]# ssh-keygen

Use ssh-copy-id command to copy public key of Ansible server to its nodes.

#### Step:4 Define the nodes or inventory of servers for Ansible.

File ‘**/etc/ansible/hosts**‘ maintains the inventory of servers for Ansible.

[root@ansible ~]# vi /etc/ansible/hosts

[test-servers]

192.168.1.9

192.168.1.10

Save and exit the file.

Sample output of hosts file.

#### Step:5 Now try to run the Commands from Ansible Server.

Check the connectivity of ‘test-servers’ or ansible nodes using ping

root@ansible ~]# ansible -m ping 'test-servers'

[root@client ~]# ansible -m ping 192.168.10.193

192.168.10.193 | UNREACHABLE! => {

"changed": false,

"msg": "Failed to connect to the host via ssh: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).\r\n",

"unreachable": true

}

This error because we need to mention user name correctly otherwise we will get error like this

#ansible -u vinod -m ping 192.168.10.193

192.168.10.193 | SUCCESS => {

"changed": false,

"ping": "pong"

}

This successful test

**Example :1 Check the uptime of Ansible nodes**

[root@ansible ~]# ansible -m command -a "uptime" 'test-servers'

[root@client ~]# ansible -m command -a "uptime" -u vinod '192.168.10.193'

192.168.10.193 | SUCCESS | rc=0 >>

14:54:56 up 3 days, 21:28, 3 users, load average: 0.00, 0.01, 0.05

**Example:2 Check Kernel Version of nodes**

[root@ansible ~]# ansible -m command -a "uname -r" 'test-servers'

[root@client ~]# ansible -m command -a "uname -r" 'test-servers'

vinod@192.168.10.193 | SUCCESS | rc=0 >>

3.10.0-514.16.1.el7.x86\_64

**Example:3 Adding a user to the nodes**

root@ansible ~]# ansible -m command -a "useradd mark" 'test-servers'

[root@ansible ~]# ansible -m command -a "grep mark /etc/passwd" 'test-servers'

[root@client ~]# ansible -m command -a "grep vinod /etc/passwd" 'test-servers'

vinod@192.168.10.193 | SUCCESS | rc=0 >>

vinod:x:1000:1000::/home/vinod:/bin/bash

**Example:4 Redirecting the output of command to a file**

[root@ansible ~]# ansible -m command -a "df -Th" 'test-servers' > /tmp/command-output.txt

If we need the output to any file we can redirect as below.

# ansible -m command -a "df -h" web-servers > /tmp/df\_outpur.txt

**Palybooks with examples create a palybook file that will copy from githib to local repository**

Cd /etc/ansible/roles/copy.yml

---

# These tasks are responsible for copying the latest dev/production code from

# the version control system.

- name: Copy the code from repository

git: repo={{ repository }} dest=/var/www/html/

- name: Creates the index.php file

template: src=index.php.j2 dest=/var/www/html/index.php

**how to install the httpd with dependencies and allow iptables and set selinux with ansible-playbooks**

---

# These tasks install http and the php modules.

- name: Install http and php etc

yum: name={{ item }} state=present ---for multiple packages to install

with\_items:

- httpd

- php

- php-mysql

- git

- libsemanage-python

- libselinux-python

- name: insert iptables rule for httpd

lineinfile: dest=/etc/sysconfig/iptables create=yes state=present regexp="{{ httpd\_port }}" insertafter="^:OUTPUT "

line="-A INPUT -p tcp --dport {{ httpd\_port }} -j ACCEPT"

notify: restart iptables

- name: http service state

service: name=httpd state=started enabled=yes

- name: Configure SELinux to allow httpd to connect to remote database

seboolean: name=httpd\_can\_network\_connect\_db state=true persistent=yes

when: sestatus.rc != 0

remote\_user=root --this is to give username

-hosts:web-servers

-tasks:

Yum: name=httpd state=installed (or) present enabled=yes

Service: name=httpd state=started (or) stopped

**If it is normal user giving sudo permission**

Hosts: web-servers

Sudo: yes

## Basics

### Hosts and Users

For each play in a playbook, you get to choose which machines in your infrastructure to target and what remote user to complete the steps (called tasks) as.

The hosts line is a list of one or more groups or host patterns, separated by colons, as described in the [Patterns](http://docs.ansible.com/ansible/intro_patterns.html) documentation. The remote\_user is just the name of the user account:

**Support for running things as another user is also available using become method**

p---

- hosts: webservers

remote\_user: yourname

become: yes

**You can also use become on a particular task instead of the whole play:**

---

- hosts: webservers

remote\_user: yourname

tasks:

- service: name=nginx state=started

become: yes

become\_method: sudo

If you need to specify a password to sudo, run ansible-playbook with --ask-become-pass or when using the old sudo syntax --ask-sudo-pass(-K). If you run a become playbook and the playbook seems to hang, it’s probably stuck at the privilege escalation prompt. Just Control-C to kill it and run it again adding the appropriate password.

When using become\_user to a user other than root, the module arguments are briefly written into a tempfile in /tmp. These are deleted immediately after the command is executed. This only occurs when changing privileges from a user like ‘bob’ to ‘timmy’, not when going from ‘bob’ to ‘root’, or logging in directly as ‘bob’ or ‘root’. If it concerns you that this data is briefly readable (not writable), avoid transferring unencrypted passwords with become\_user set. In other cases, /tmp is not used and this does not come into play. Ansible also takes care to not log password parameters.

The **command** and **shell** modules are the only modules that just take a list of arguments and don’t use the key=value form. This makes them work as simply as you would expect:

tasks:

- name: enable selinux

command: /sbin/setenforce 1

The **command** and **shell** module care about return codes, so if you have a command whose successful exit code is not zero, you may wish to do this:

tasks:

- name: run this command and ignore the result

shell: /usr/bin/somecommand

ignore\_errors: True

To check the syntax of a playbook, use ansible-playbook with the --syntax-check flag. This will run the playbook file through the parser to ensure its included files, roles, etc. have no syntax problems.