**Ansible**

Configuration Management: This is the process of provisioning handling the changes in a system so as to maintain integrity and consistency

* Server configuration management can be done by tools like Ansible, Chef, and puppet
* These tools help us in automating the server configuration

Example: If there are 1000 servers we want to deploy a patch on all these servers which are present at different geographical locations. We can use any of the above discussed Configuration Management (CM) tools.

Using CM tools it become possible to control multiple servers from one point of control

**Advantages:**

* **Quick Provisioning**: Whenever a new server has to be deployed CM tools can automate most of the process
* **Quick Recovery from critical events**: Whenever a sever goes offline due to some unknown reasons it might take hours to properly audit the system and perform RCA (root cause analysis). CM tools are used for deploying replacement servers very quickly which can replace the original server for time being
* **Efficient Handling of Snowflakes servers**: Multiple servers can be working on different softwares or different versions of softwares Example one server has Git 2.1 installed on it another server has Git 1.7 and Apache 2 installed on it and another server has FTP and Jenkins installed on it. Instead of maintaining heavy documents to handle these servers we can use CM tolls like Ansible and perform these actions using a single script
* **Replicated Environment**: Snap shot of a complete data centre can be captures using CM tools and using this snap shot it becomes very easy to create another data centre with the same configuration
* **Idempotent Behaviour**: CM tools implement idempotent policy that is if a particular server already has the configurations changes that Ansible is trying to do then Ansible will not disturb that server Example: If there are 1000 servers on which we want to install Git 2.7 and already on 500 servers Git 2.7 is installed then Ansible will not disturb them and on the rest 500 servers Ansible will install Git 2.7
* **System Facts**: CM tools capture detail information about the system that is been provisioned or configures currently and store it in some variables. These are called system facts. Using these system facts it becomes easy to create similar setups on different machines
* **Templating System**: CM tools can create automation scripts which do not work on hard coted values instead they can work on different sets of values which are provided on run time Example: We can create an Ansible playbook for installing some services which service has to be installed can be decided on run time that is using the same playbook one time we can install Jenkins and another time we can install Git

Difference between Ansible, Puppet and Chef

|  |  |  |  |
| --- | --- | --- | --- |
| Description | Ansible | Puppet | Chef |
| Script Language | YAML | Custom DSL based Ruby | Ruby |
| Infrastructure | controller machine applies configuration on an nodes via ssh | Puppet master synchronize configuration on Puppet nodes | Chef workstation push configuration on to chef server from which chef nodes will be updated |
| Requires specialized software for nodes | No | Yes | Yes |
| provide centralized point of control | No, any computer can be a controller | Yes, via Puppet Master | Yes, via Chef Server |
| Script terminology | Playbook / Roles | Modules / Manifest | Recipes / Cookbook |
| Task execution order | Sequential | Non - Sequential | Sequential |

Creating infrastructure for Ansible:

* Download and install oracle virtual box.
* Download and install vagrant.
* Copy vagrant file into a specific folder.
* Open cmd prompt
* Cd path\_of\_folder\_vagrant\_file\_is\_copied
* # vagrant up.

Ansible:

Ansible is an open source configuration management and archest ration utility. It can automate and standardize the configuration of remote host and virtual machines.

Ansible can be used to co-ordinate the launch and grace full shutdown of multitired applications. Due to this reason Ansible is used to performing rolling updates of multiple systems with zero down time.

Ansible was created using python and it uses a concept called plays for performing system administration activities.

Play is a series of tasks that can be performed on servers.

The file which contain multiple plays called as playbook.

\*\* Ansible follows an agent less architecture. Work is pushed to remote host via SSH.

DISSADVANTAGES.

Ansible can be used for installing softwares on remote machine but it cannot perform the basic installation of operating systems.

Ansible cannot monitor the changes done on remote machine.

Ansible requires python to be installed on the controller machine and also the host machine.

Note:

Controller:

This is the main machine where we install Ansible.

Managed nodes or hosts:

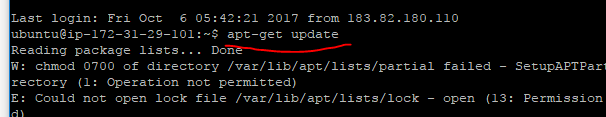
These are the remote machines which will be controlled by through Ansible.

Ansible installation:

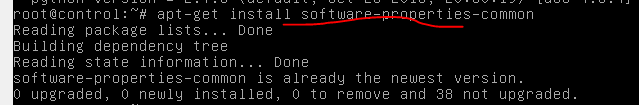
* Open oracle virtual box go to control VM.
* Enter username and password as vagrant and vagrant.

Commands for installing Ansible:

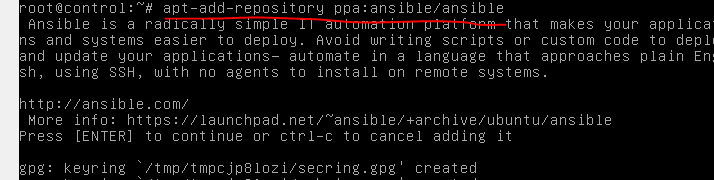
* $ sudo apt-get update



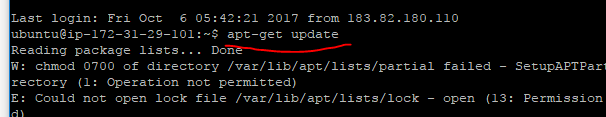
$ sudo apt-get install software-properties-common



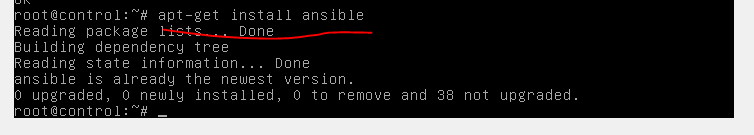
* $ sudo apt-add-repository ppa:ansible/Ansible



* $ sudo apt-get update

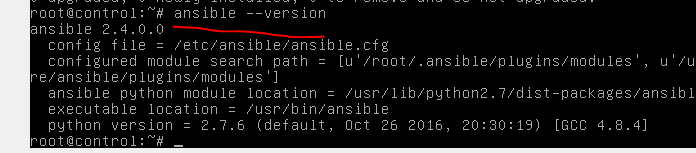


* $ sudo apt-get install Ansible.



Check the version of Ansible installed on machine.

# Ansible --version



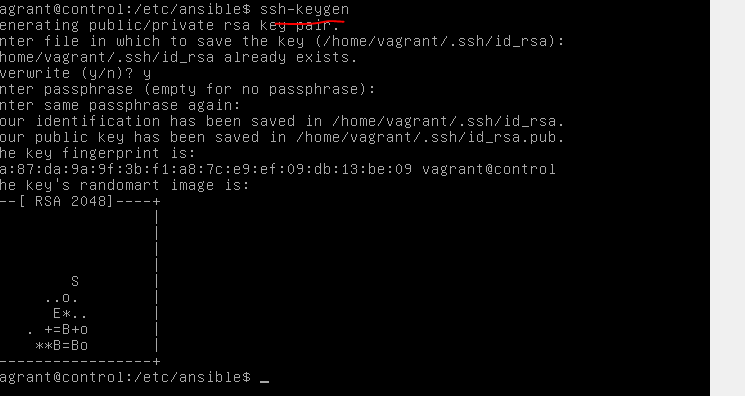
10.10.2017

Ansible comminucates with the remote machines via SSH. This should be password less

STEPS:

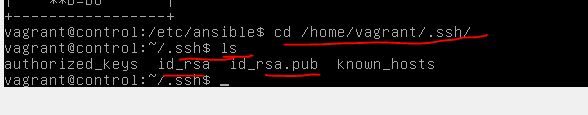
To genetare SSh keys

# ssh-keygen



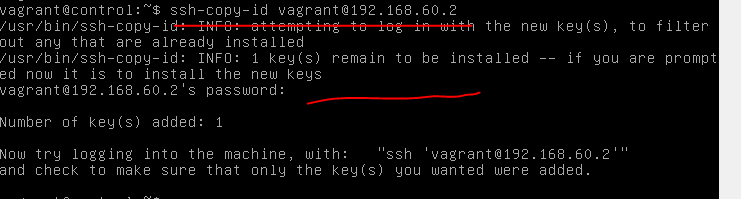
The above command will generate 2 keys

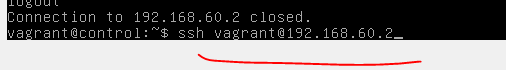
* id\_rsa
* id\_rsa.pub

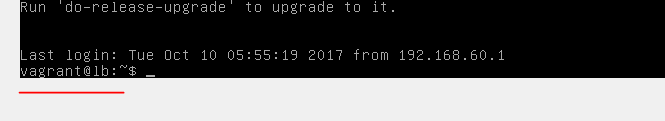


The public keys should be copied to the remote machines

# ssh-copy\_id username@ipaddress\_of\_remote\_machine







Similarly repeat the above process of connecting password less authentication with remaining servers

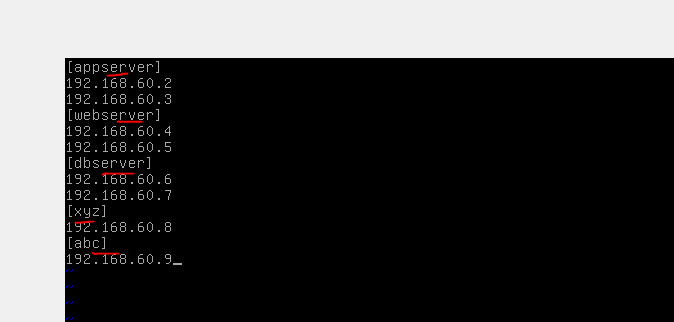
This will copy the contents of id\_rsa\_pub into a file called “authorizedkeys” which is present in .ssh folder at remote machine without providing password ssh username@ipaddress



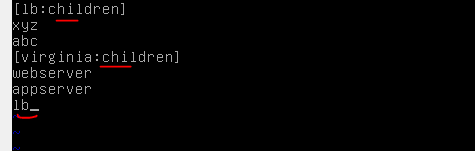
Ansible Inventory File: Ansible performs operations on the remote servers whose information is present in the inventory file. This file is present on the controller machine in /etc/Ansible folder. The name of the file is **hosts.** In this file we should specify the IP address of all the managed nodes

Grouping of servers:

Collection of remote hosts can be grouped under one name in the inventory file by specifying in square brackets [ ]



Grouping of group can be done using the children key word



E.g.: [lb: children]

Xyz

Abc

[Virginia: children]

Webserver

Appserver

Lb

Whenever we ask Ansible to perform any activity in lb group it will perform on both xyz and abc group. Similarly when we ask Ansible to perform any activity on Virginia group it will perform on webserver, appserver and lb (lb includes abc and xyz groups)

Ansible Modules: These are inbounded modules which have been created in python and Ansible uses them for performing different activities on managed nodes

Important Ansible modules:

* Command Module: This sis the default module of Ansible and it is used for running Linux commands on managed nodes.
* Shell Module: This sis used for running SHELL commands on managed nodes

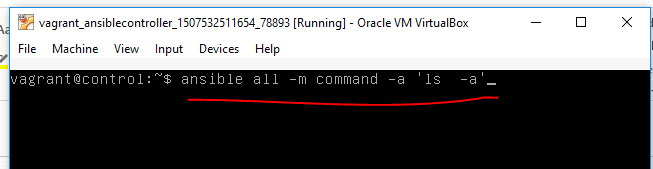
e.g.: directions (>>) and piping (|)

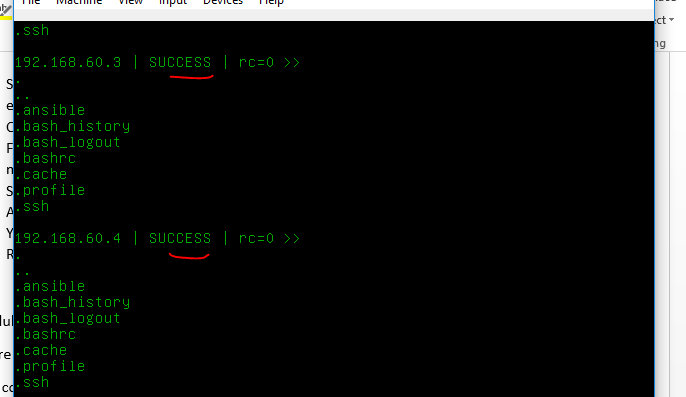
* Copy Module: This is used for copying files and directions into the managed nodes
* Fetch Module: This is used for capturing files from managed nodes into controller machine
* STATS Modules: This is used to display statistics of the managed nodes
* Apt module: This is used for package management in Ubuntu machines
* Yum module: This is also a package management software like apt but it works on RED HAT Linux, Centos, Oracle Enterprise Linux etc.

Command Modules: We can use it for executing Linux commands on managed nodes

Command to fire ls -a on all the managed nodes

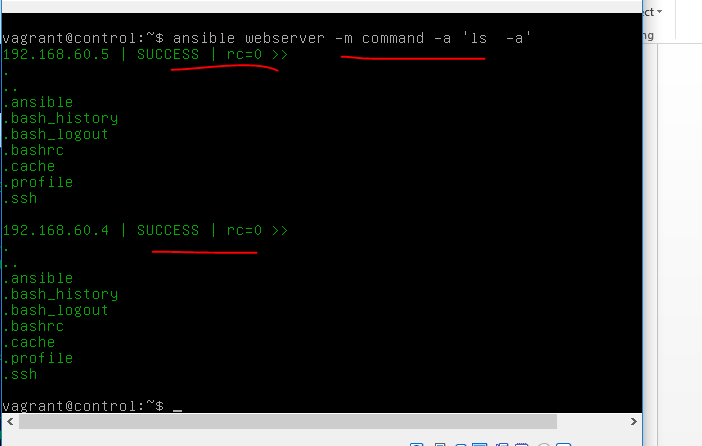
# ansibe all –m command –a ‘ls -a’





To execute the same command on a specific group called webserver present in our inventory file

# Ansible webserver –m command –a ‘ls -a’

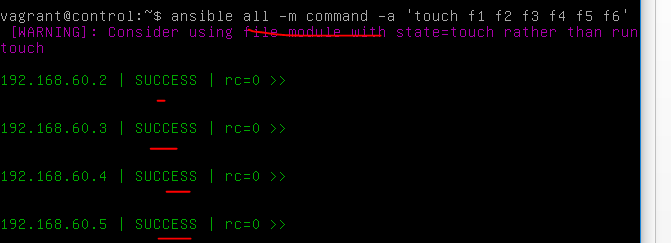


To execute the same command on one single managed node

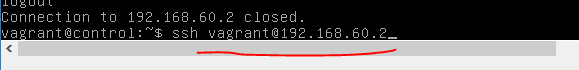


To create 6 files using touch command om all the managed nodes

# Ansible all –m command –a ‘touch f1 f2 f3 f4 f5 f6’



For confirmation

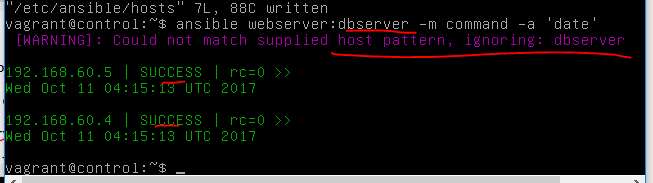




Union intersection and inverse of groups: It can be done using : , :& and :!

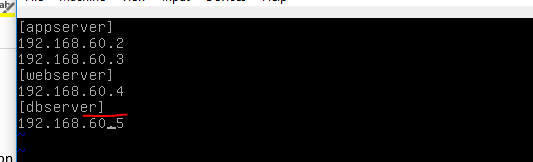
To perform date command on all the servers present in webserver group and dbserver group

# Ansible webserver:dbserver –m command -a ‘date’

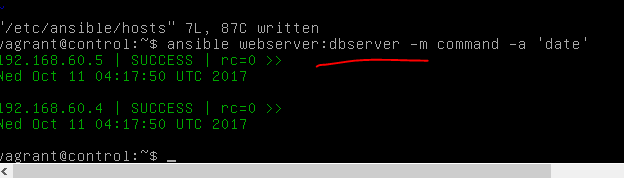


Db server is not available in inventory file so Ansible is ignoring that server and performing on rest of the server that webserver.

So we are creating db server in default hosts file



Success



The above command will perform a union operation on two groups

Union means if a server is available in different groups then with the above command activity can be done and Ansible will perform only once by selecting that single server from different groups

To perform date command on managed nodes which are present in both webserver and dbserver

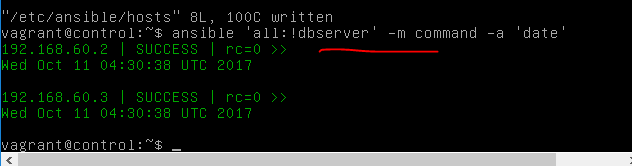
# Ansible ‘webserver:&dbserver’ -m command –a ‘date’



The above command will perform intersection operation

To execute on all the servers except a group called dbserver

# ansible ‘all:!dbserver’ –m command –a ‘date’

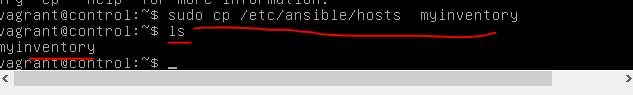


By default ansible reads inventory information from /etc/ansible/hosts onstead we can create our own inventory file ans ask ansible to read from that file

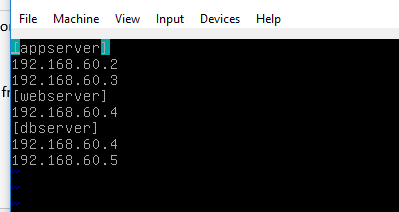
Steps:

Copy hosts from of /etc/ansible/hosts to a file called myinventory

# sudo cp /etc/ansible/hosts myinventory





Add managed hosts IP’S address to that fiel (myinventory)

To call our own inventory file (myinventory)

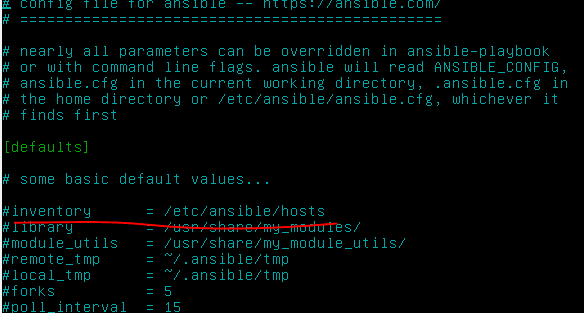
# ansible –i myinventory webserver –m command –a ‘date’



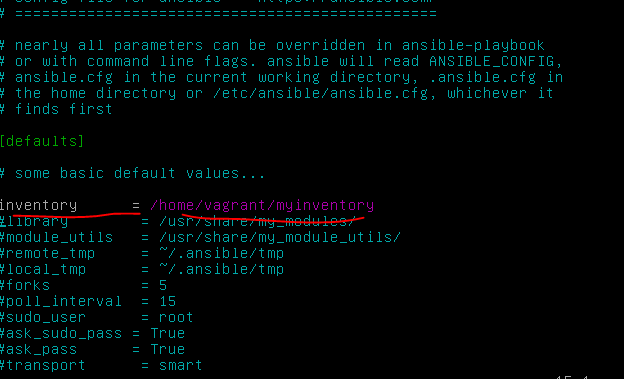
The default confifguration file inforamtion can be modified by openning ansible.cfg file which is present in /etc/ansible folder

# sudo vim /etc/ansible/ansible.cfg

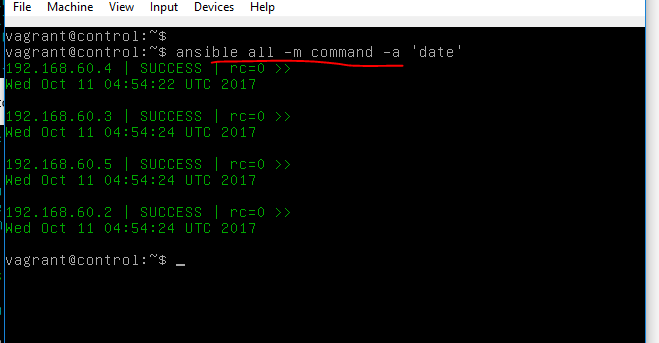




Go to inventory varable remove # (uncommand) & specify the path og our inventory file



Absible is reading my inventory file (MYINVENTORY) created and updated in default location of ansible

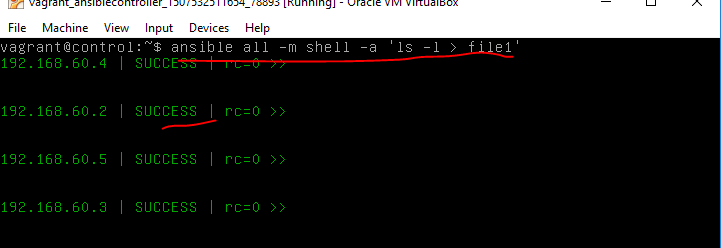


12.10.2017

Shell Moduel: We can use this module for executing commands which invloce redirections and piping

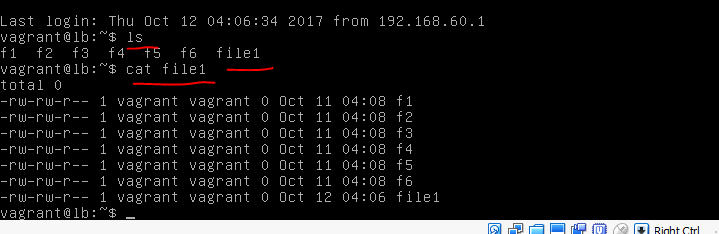
Q. Store the output of ls –la into a file called file1 on all managed notes

# ansble all –m shell –a ‘ls -la > file1’



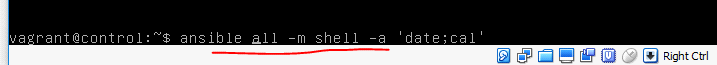
For output log in to anu managed node

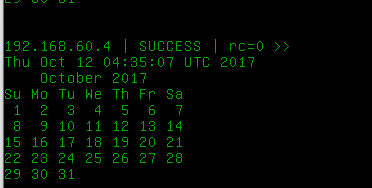




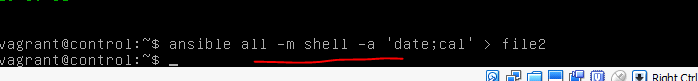
Q: To execute multiple commands by separting them with a semi colun (;) we should use shell module command to execute cal and date on all the managed nodes

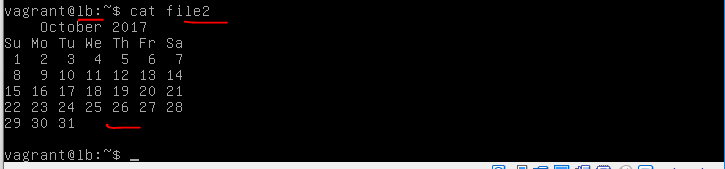
# ansible all –m shell –a ‘date;cal’





Only last command is executed(cal) and stored in file2

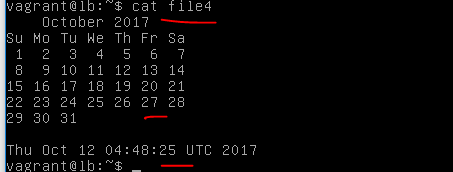




In order to exeute both command and stores in file2

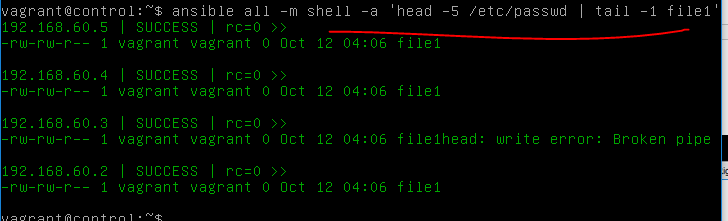


For Output log to managed nodes and open file4



Q: Capture the 5th line from /etc/passwd file and stored it in a file called file1 on all the managed nodes

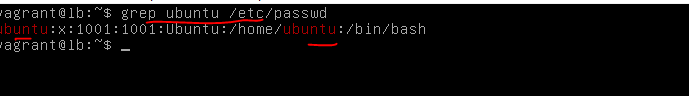
# ansible all –m shell –a ‘head -5 /etc/passwd | tail -1 file1’

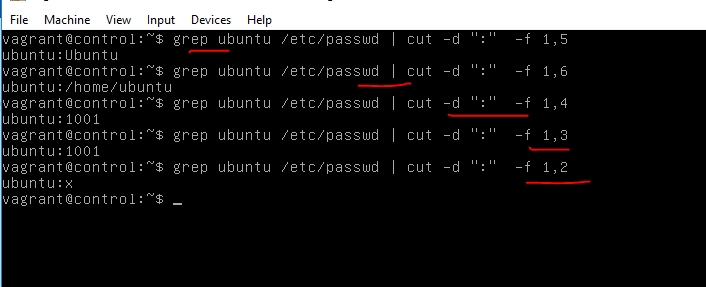
for output log on to nodes and check

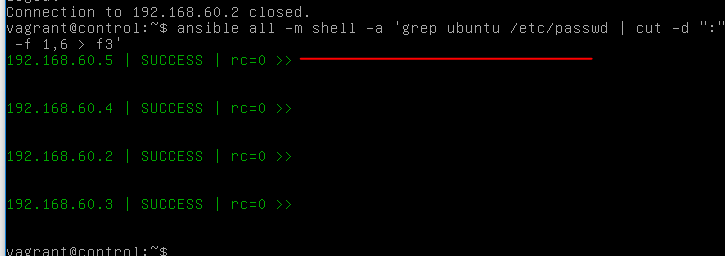


Q: From /et/passwd file caoture all the lines where the word ubuntu is present and from this data capture only 1st column and 6th column and store it in file called file1 on all amanged nodes

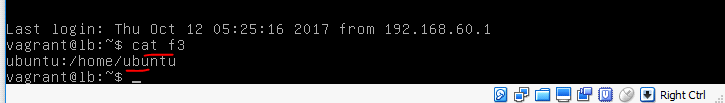
# ansible all –m shell –a ‘grep ubuntu /etc/passwd | cut –d “:” –f 1,6 > file1







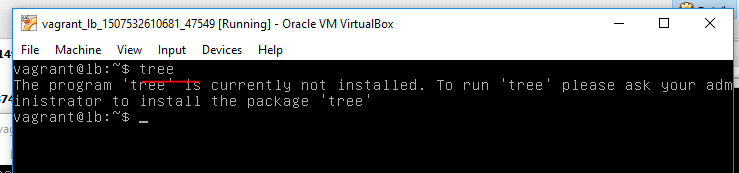
Output



13.10.2017

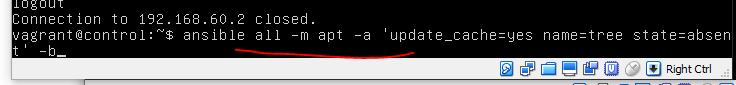
Package Management using Ansible: we can do this using either the apt module or yum module. In ununtu, debain etc machines we use apt where as in Red Hat Linux, fedora, centos oracle emterprises Linux we use yum. Using apt or yum it is possible to install packages or uninstall packages or upgrade them etc

Tree is not install on managed nodes



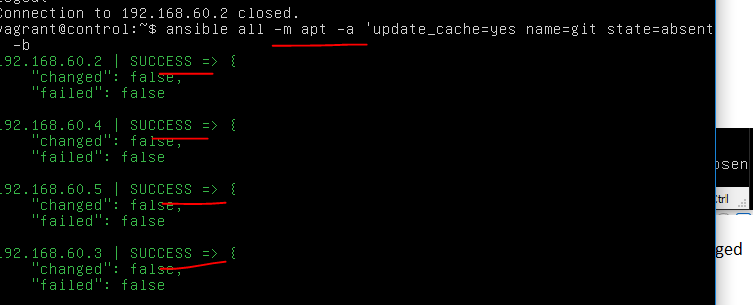
Q: Write a anisble command for installing tree package on all managed nodes

# ansible all –m apt –a ‘name=tree state=present’ -b



Q: Write ansible adoc command to update apt repository and then install git on all managed nodes

# ansible all -m apt -a 'update\_cache=yes name=git state=present' -b



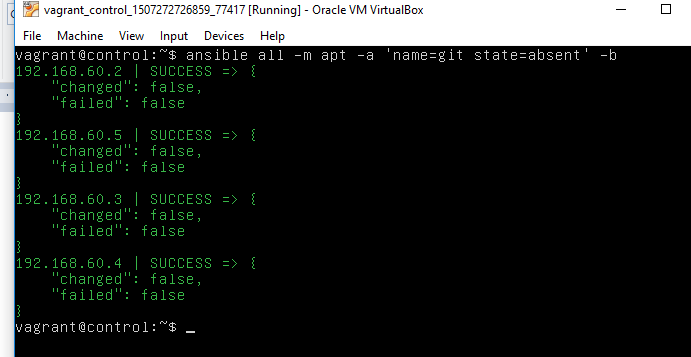
update\_catch = yes is used for upgrading the apt repository

-b stands for becoming and it is used for privilege escalation that is it gives temporary root access on all the managed nodes to install or uninstall packages

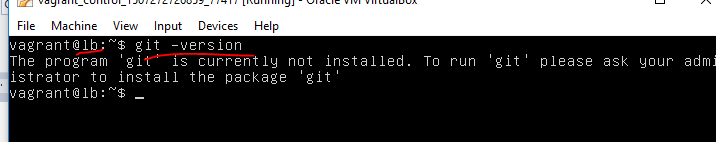
Similarly we can give state=absent to uninstall the package from the managed nodes

Q: Write an anisble command to uninstall git from all the managed nodes

# ansible all -m apt -a 'name=git state=absent -b



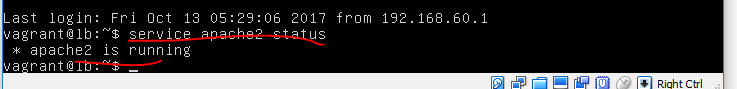
for confirmation log on any node ssh vagrant@192.168.60.2 and check the result



Q: Write a ansible adhoc command to install apache2 on all managed nodes

# ansible all -m apt -a 'name=apache2 state=present' -b

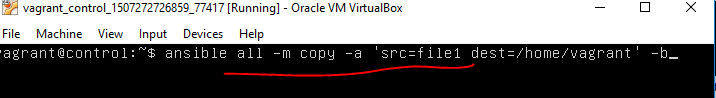




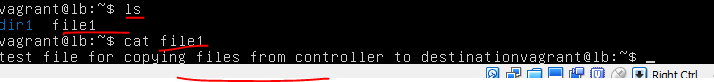
Copy Modules: This is used to copy files or directories into all the managed nodes

Q: Write an ansible command to copy a file from the controller machine into all the managed nodes

# ansible all -m copy -a 'src=file1 dest=/home/vagrant

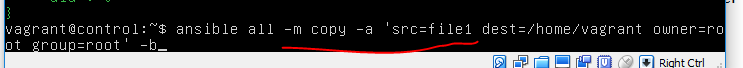


For output log on to any managed nodes and check



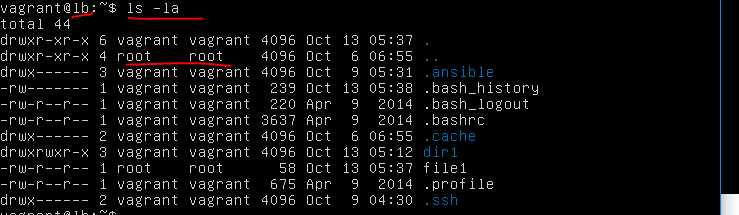
Q: Write a ansible command to copy a file to all the managed nodes and change the owner ship & owner group

# ansible all -m copy -a 'src=file1 dest=/home/vagrant' owner=root group=root' -b



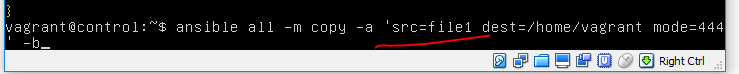


output



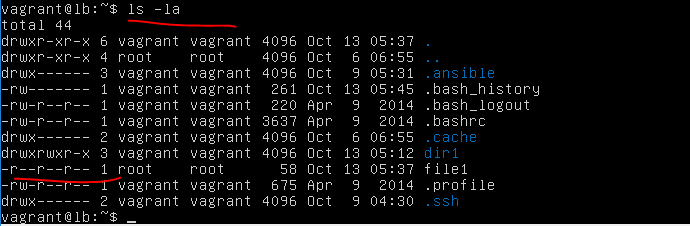
Q: Write an ansible command to copy a file to all the managed nodes to set only read permissions on file for owner group and other

# ansible all -m copy -a 'src=file1 dest=/home/vagrant mode=444' -b





Output



14.10.2017

Copy command can also be used to copy the complete directory along with its sub directories and files

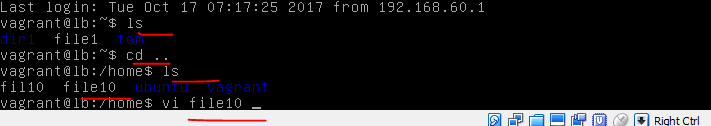
# Ansible all –m copy –a ‘src=dir1 dest=/home/vagrant’

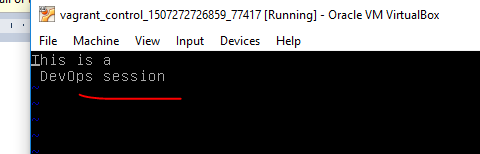
We can also send the content into directory using copy module

# Ansible all –m copy –a ‘content=”this is a \n devops session \n” dest=/home/file10’



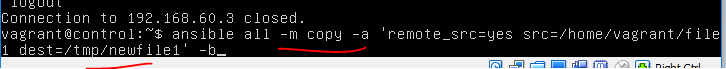
For ouput log on to any node and check it



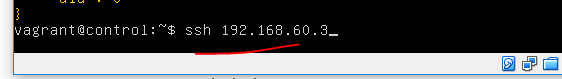


It also possible to copy a file present on the managed node to some other folder managed nodes this can be done argument remote src=yes

# Ansible all –m copy –a ‘remote\_src=yes src=/home/vagrant/file1 dest=/tmp/file1’ –b



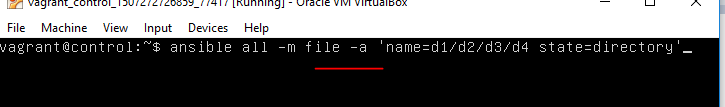
For Output



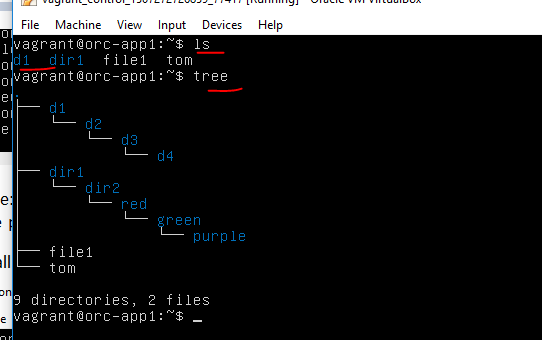


File Module: This is used for creating files or directories on the managed nodes. We can also change the permissions, ownership and group ownership etc.

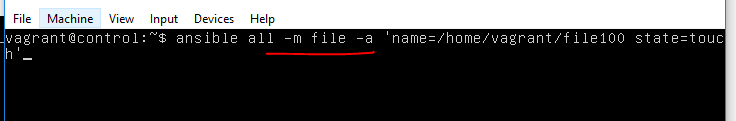
# Ansible all –m file –a ‘name=d1/d2 state=directory’



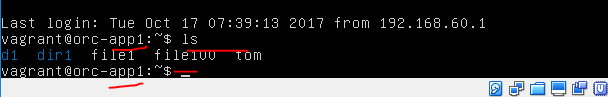
For output log on to any node



# Ansible all –m file –a ‘name=/home/vagrant/file1 state=touch’



success



We can also change the permissions which create on remote machines

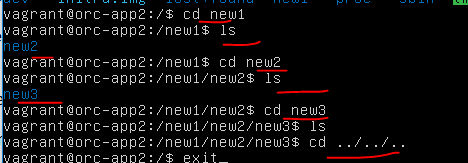
# Ansible all –m file –a ‘name=/home/vagrant/file11 state=touch chmod=777’ –b

# Ansible all –m file –a ‘name=/home/vagrant/file11 state=touch chowner=root group=root’ –b

To create a directory state = directory

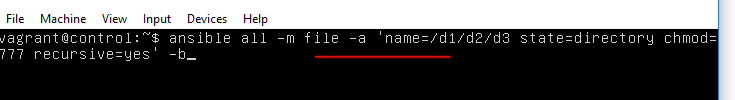
# Ansible all –m file –a ‘name=/new1/new2/new3 state=directory’ –b

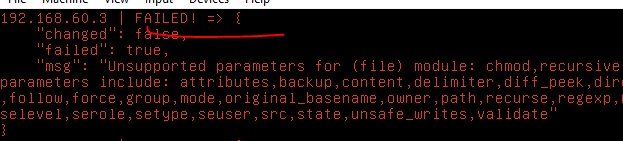




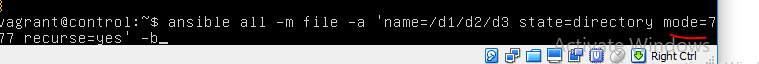
To recursively change the permissions

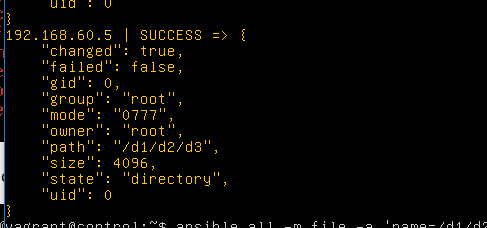
# Ansible all –m file-a ‘name=/d1/d2/d3 state=directory chmod=777 recursive=yes’ –b



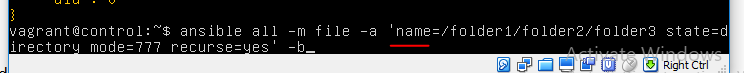


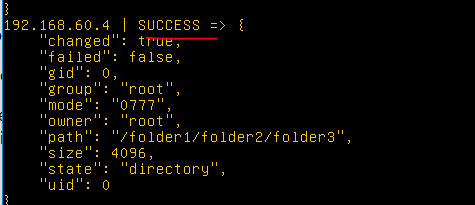
change chmod to mode



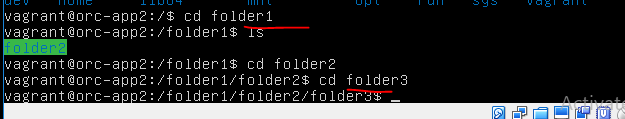


# Ansible all –m file-a ‘name=/folder1/folder2/folder3 state=directory mode=777 recurse=yes’ –b

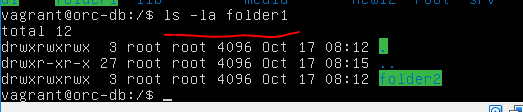




folder is created in nodes



user and group of the folder is also changed

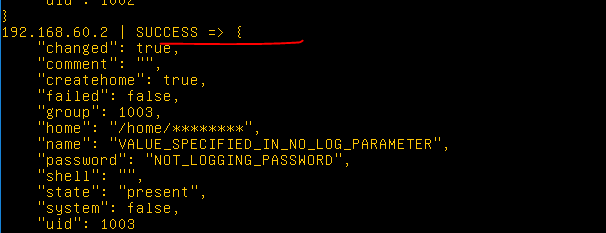


User Module: This module is used for creating the users. It can also use for setting password

To create a user and assign a password

# Ansible all –m user –a ‘name=sai password=intel’ –b



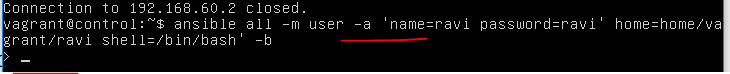


output



To create a user assign home directory and default shell

# Ansible all –m user –a ‘name=ravi password=intel’ home=/home/vagrant/ravi shell=/bin/bash’ –b



mistake with ansible adhoc

16/10/2017

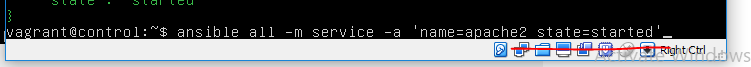
Service Module: This module is used for starting or stopping services on managed notes

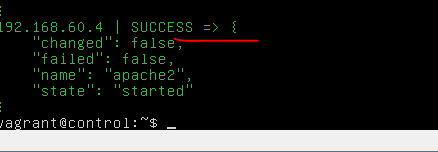
Install apache2 on all managed nodes using apt module and stop that services using service module

Installing apache2

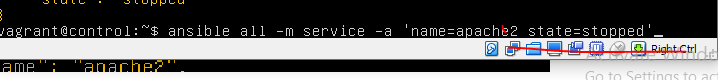
# anisble all –m apt –a ‘name=apache2 state=present’ -b

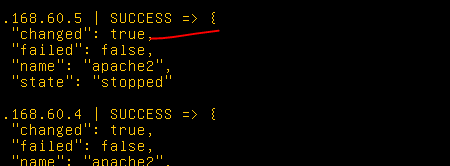
starting apache2 service





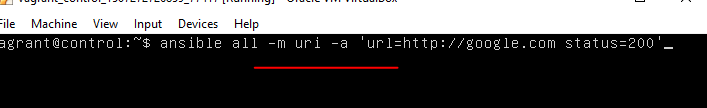
Stopping apache2



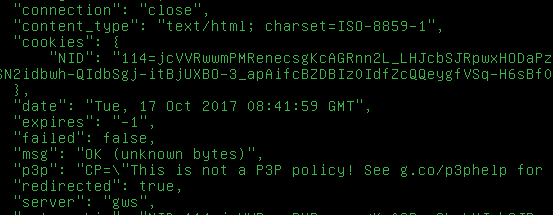


uri module: This is used to check if a particular site or web application is accessible from a remote managed node. If it is accessible it will return he exit port as 200 if it is not accessible it will return the exit port as 401 or 404 etc.

# Ansible all –m uri –a ‘url=http://google.com status=200’



Success

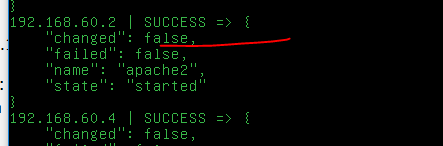


When we start apache2 on any managed nodes in /var folder it will create another folder called www. This folder contains another folder html. This is the default webpage present in appache2. To check whether this page is reachable on that managed nodes

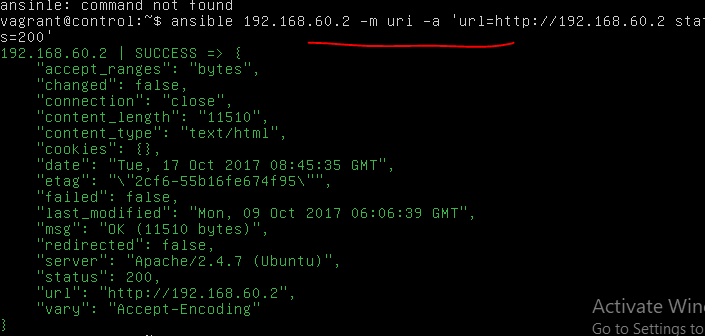
Start apache2 on the managed node using service module check whether the default page is reachable using uri module

# Ansible 192.168.60.102 –m service –a ‘name=apache2 state=started’ –b





# Ansible 192.168.60.102 –m uri –a ‘url=http://192.168.60.102 status=200’



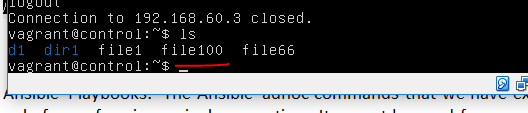
Fetch Module: This is used for copying files which are present on managed nodes and bringing them into the controller machine

* Create a file in any one managed node and fetch that file into the controller machine
* # Ansible 192.168.60.102 –m fetch –a ‘src=/home/vagrant/file1 dest=file1 flat=true’





ouput



Ansible Playbooks: The Ansible adhoc commands that we have executed till now can be used only for performing a single operation. It cannot be used for complex configuration management operations.

Adhoc co0mmands can invoke one module at a time and it can work only on one set arguments

To work on multiple modules we should use Ansible playbooks

Playbooks are the files created in yaml format. They offer powerful and flexible solutions for configuration management

Using playbooks it is possible to change lengthy and complex administrative tasks into easily repetitive solutions

In Playbook each play defines a set of operations that can be performed on managed nodes. These operations are called as tasks and the managed nodes are called hosts.

Each tasks invokes specific modules. The tasks are executed in the order that they are specified in the yaml file.

The advantage of playbooks in system admin can configure a playbook which can be used by developers or testers who do not have knowledge on configuration management but still they can perform activity on managed nodes.

To check syntactical errors we use following command

# ansible-playbook –syntax-check playbook\_name.yml

1. Create an ansible playbook for creating directory on all the managed nodes and then copying file into it.

* To perform this activity using adhoc commands. We should execute two different modules separately. 1. File module 2. Copy module

1. # ansible all –m file –a ‘name=newdir1 state=directory’
2. # ansible all –m copy –a ‘src=/home/vagrant/newfile1 dest=/home/vagrant/copiedfile

(OR)

We can create playbook with those 2 commands in a single playbook.

# sudo vim playbook.yml

* Name: creating directories and copying files

Hosts: all

Tasks:

- name: creating directory

File:

name: newdir1

state: directory

* Name: copying files

Copy:

Src: /home/vagrant/newfile1

Dest: /home/vagrant/copied file

…

1. Create an ansible playbook for creating the directory on all the managed nodes. In that directory create a file and then send some sample data into a file.

# sudo vim playbook2.yml

* Name: creating directories and creating files and sending content

Hosts: all

Tasks:

- name: creating directory

File:

Name: newdir2

State: directory

- name: creating file

File:

Name: file1

State: touch

- name: sending content

Copy:

Content: “sample data\n”

Dest: newdir2/file1

….

1. Create a playbook for capturing all the users from /etc/passwd file who are using shell as /bin/false. Stores this data in a file on the managed nodes and fetch those files into the controller machine.

# sudo vim playbook3.yml

* Name: capturing some content from /etc/passwd and save in a file and fetch file into controller

Hosts: all

Tasks:

- name: capturing data from /etc/passwd and store in a file1

Shell: grep /bin/false /etc/passwd > file1

- name: fetching files

Fetch:

Src: /home/vagrant/file1

Dest: /home/vagrant

…

1. Create an ansible playbook for creating users on all the managed nodes. Capture that user information from /etc/passwd file and capture only username and home directory store it in a file and fetch that file.

# sudo vim playbook4.yml

- name : creating users and fetching their information

Hosts: all

Become: yes

Tasks:

- name: creating users

User:

Name: shafi

Home: /home/vagrant/shafi

Shell: /bin/bash

- name: capturing username and home directories

Shell: grep shafi /etc/passwd | cut –d “:” –f 1,6 > file

- name: fetching files

Fetch:

Src: /home/vagrant/file

Dest: /home/vagrant

….

1. Create an ansible playbook for installing apache2, starting apache2 service and checking the url response.

Sudo vim playbook5.yml

- name: installing apache2, starting it and checking url response

Hosts: all

Tasks:

- name: installing apahce2

Apt:

Name: apach2

State: present

Update\_cache: yes

- name: starting apahce2 service

Service:

Name: apache2

State: started

- name: checking url response

Uri:

url: <http://ip> address

status: 200

…..

Note:

1. Ansible playbook can also be created to work on multiple host group and different kind of activities.
2. That is on one group of host it will perform one kind of activity and another group it will perform another activity.
3. Create an ansible playbook for installing apache2 on webserver group and git on lb group.

Sudo vim playbook6.yml

- name: installing apache2 on web server group

Hosts: webserver

Tasks:

- name: installing apache2

Apt:

Name: apache2

State: present

Update\_cache: yes

- name: installing git on lb group

Hosts: lb

Tasks:

name: git

state: present

update\_cache: yes

….

VARIABLES IN ANSIBLE:

Variables are categorized into 3 types

1. Global scope variable
2. Play scope variable
3. Host scope variable
4. Global scope variable:

G.S variables can affect the complete playbook. They are defined from command prompts.

They have the highest precedence.

1. Play scope variable

P.S variables can impact only one play in which they are defined.

1. Host scope variable

H.S variables can impact host or group of host.

7. Create an ansible playbook which can be used for installing or uninstalling packages. It should also be used for either updating or not updating apt repositrory.

Sudo vim playbook7.yml

- name: using global variables

Hosts: all

Tasks:

- name: Installing or uninstalling packages

Apt:

Name: “{{a}}”

State: “{{b}}”

Update\_Cache: “{{c}}”

…

Using playbook to install apache2 after updating repository.

Ansible –playbook playbook7.yml –extra-vars “a=apache2 b= present c=yes” –b

We can also use the same playbook for uninstalling any packages.

To uninstall git without updating repository.

Ansible-playbook playbook7.yml –extra-vars “a=git b=absent c=no” –b

1. Create an ansible playbook which can be used for creating files or directories, controlling owner and group owner and specifying permissions.

All above should control from command prompt.

Sudo vim playbook8.yml

- name: creating files or directories

File:

Name: ‘{{a}}”

State: “{{b}}”

Owner: “{{c}}”

Group: “{{d}}”

Mode: “{{e}}”

Ansible-playbook playbook8.yml –extra-vars “a=file1 b=touch c=root d=vagrant e=765” –b

2. Play Scope variables:

These variables can impact only one play.

Create a playbook for installing apache after updating apt repository using play scope variables.

Sudo vim playbook10.yml

- name: package installation or uninstallation

Hosts: all

Vars:

- a: apache2

- b: present

- c: yes

Tasks:

- name: installing packages

Apt:

Name: “{{a}}”

State: “{{b}}”

Update\_cache: “{{c}}”

The above playbook will by default install apache2. But we can user the same playbook for installing or uninstalling other services by sending these volumes from command prompt.

The values sent from cmd prompt are global scope and they have more priority compare to play scope variables.

To use the same play book to uninstall git without updating repository.

Ansible-playbook playbook10.yml –extra-vars “a=git b=absent c=no” –b

3. Host Scope variables:

These variables are used to affect a specific group of host or individual host.

**To work on group of hosts:**

For this we should create a directory “group\_vars”.

This directory should be created in same folder where the playbooks are present.

1. Mkdir group\_vars
2. Cd group\_vars
3. Cat > web server(group name)

Create a file with the name of group and in this file defined variables.

1. A: maven
2. B: present
3. C: yes
4. Save the file
5. Come back to the directory where one playbooks are present i.e., cd..
6. Create playbook for installing maven with apt on web server group.

Sudo vim playbook11.yml

- name: package installation or uninstallation

Hosts: webserver

Tasks:

- name: installing packages

Apt:

Name:”{{a}}”

State: “{{b}}”

Update\_cache: “{{c}}”

To create host level variables for individual machine

1. Create folder called host\_vars
2. In the same folder where the playbooks are present
3. Mkdir host\_vars
4. Crate a file with the name of ip address of host machine
5. Cat > 192.168.60.101
6. User: shafi
7. Home: /home/vagrant/shafi
8. Passwd: intelliq
9. Save the file come back to playbook folder i.e., cd ..
10. Create a ansible playbook for creating user assigning home directory and password on the managed nodes.

Sudo vim playbook12.yml

- name: Creating users

Hosts: 192.168.60.102

Tasks:

- name: user creation and home directory

User:

Name: “{{user}}”

Home:”{{home}}”

Password: “{{passwd}}”