**Ansible**

Link : <http://www.mydailytutorials.com/list-ansible-topics/>

Jenkins Roles : <https://www.youtube.com/watch?v=SgxtxzQJM4w>

**Project is explained here watch 2nd half of the video.**

Ansible is a configuration management and provisioning tool, similar to Chef, Puppet or Salt. It uses SSH to connect to servers and run the configured Tasks.

  It uses a very simple language (YAML, in the form of Ansible Playbooks) that allow you to describe your automation jobs in a way that approaches plain English.

Ansible works by connecting to your nodes and pushing out small programs, called "Ansible modules" to them.  Ansible then executes these modules (over SSH by default), and removes them when finished.

Add your SSH private key to the ssh-agent. If you created your key with a different name, or if you are adding an existing key that has a different name, replace *id\_rsa* in the command with the name of your private key file.

ssh-agent bash ;

ssh-add ~/.ssh/id\_rsa

**Manage Ansible Static and Dynamic Host Inventory :**

As we know that **Ansible** is the most powerful **automation tool** that can configure the hosts at ease. The main benefit of using Ansible as a automation tools is that we don’t have to install any agent on hosts. Communication between Ansible server and its clients or managed hosts is agentless.

 The system on which we install ansible software  is called as “**Control Node**”.

The server which are managed by ansible server are called as “**Managed Host “.**

Managed Hosts entries are stored in a host inventory file, it is a text file on control node which consists of managed host name or ip addresses

In Ansible  we can manage two type hosts inventory i.e **static** and  **dynamic**.

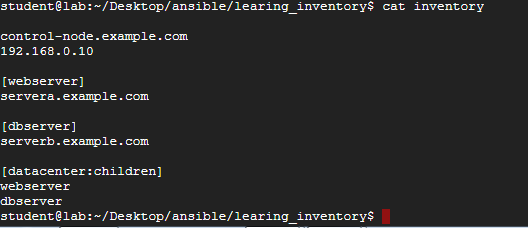
In this tutorial I will be using followings:

**One Control Node** –   control-node.example.com (192.168.0.10)

**Two Managed Hosts** – servera.example.com (192.168.0.20) and serverb.example.com (192.168.0.30)

**Static Host Inventory :**

Each section begins with a host group name enclosed in a square brackets([]) then the host entries are listed for that group.

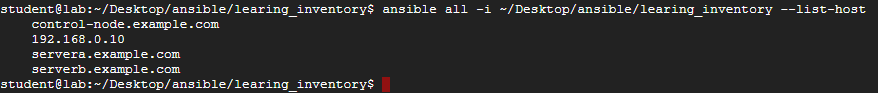


In the inventory file I have created two hosts group with the name webserver and dbserver, apart from this we have created one more group with the name datacenter that include groups of host groups. Anisble host inventories can include groups of host groups, this is accomplished with ‘**:children**‘ suffix example is shown in above created inventory. Also It is not compulsory to place host in a group we can simply place the hosts without mentioning the host group just like “control-node.exmaple.com” entry in the inventory file.

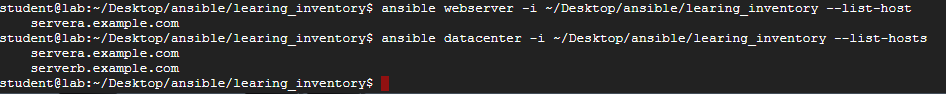
To use ansible command for host management, path of host inventory file must specified with “**-i**” option.

$ ansible {host-pattern}  -i  /<path\_of\_inventory\_file>  –list-hosts

#### **Example:1 List all the manage host :**

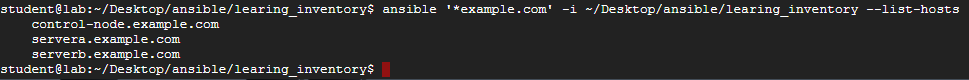


#### **Example:2 List the managed hosts based on host group.**



#### **Example: 3 List managed hosts based on wild card host pattern.**

List all the hosts which are on the domain “\*.example.com”



List all the hosts which are on network “192.168.0.0”



#### **Example:4 Advanced host pattern like inclusion and exclusion**

Apart from wildcards, Ansible allows us to create complex host patterns using inclusion and exclusion logic. Inclusion is accomplished with ‘:’ character to separate groups in host pattern to indicate an OR logic.

**Host Inclusion pattern example**



**Host Intersection pattern example**

‘:&’ represents intersection of two groups in the inventory file.

List the entries which are in both the group.



**Host Exclusion pattern example**

Exclusion is accomplished using the ‘:’ character in conjunction with the ‘!’ character.



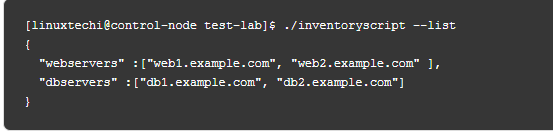
**Dynamic Host Inventory :**

Host inventory in Ansible can be dynamically generated. Sources for dynamic inventory information include public / private cloud providers.

For Cloud providers, authentication and access information should be defined  in files that script can access. A Number of existing scripts are available from Ansible’s GitHub Site at https://github.com/ansible/ansible/tree/devel/contrib/inventory, these scripts support the dynamic generation of an inventory based on host information avai;able from a large number platforms like **Openstack**, **AWS**, **Ovirt**, **Red Hat Satellite** and **OpenShift**.

We can write our own  customize dynamic inventory program in any programming language and must return in JSON format when passed appropriate options.

In order for Ansible to use script to retrieve hosts information from external inventory system, this script has to support the**–list** parameter, returning host group and hosts information similar to the JSON hash/dictionary. Example is shown below :



A Script creating a dynamic inventory has to be executable in order for Ansible to use it.

**Note :** <https://www.linuxtechi.com/manage-ansible-static-and-dynamic-host-inventory/>

**YAML Basics:**

All YAML files (regardless of their association with Ansible or not) can optionally begin with ---and end with .... This is part of the YAML format and indicates the start and end of a document.

### A dictionary is represented in a simple key: value form (the colon must be followed by a space):

*# An employee record*

martin:

name: Martin D'vloper

job: Developer

skill: Elite

You can also use abbreviation to represent dictionaries.

Eg : Martin : {name: Martin D'vloper, job: Developer, skill: Elite }

### We can also represent List in YAML. Every element(member) of list should be written in a new line with same indentation starting with “- “ (- and space).

**---**

*# A list of tasty fruits*

fruits:

- Apple

- Orange

- Strawberry

- Mango

**...**

You can also use abbreviation to represent lists.

fruits: [‘Apple’,’Orange’,’Strawberry’,’Mango’]

**List inside Dictionaries :**

---

james:

name: james john

rollNo: 34

div: B

sex: male

likes:

- maths

- physics

- english

…

### Ansible doesn’t really use these too much, but you can also specify a boolean value (true/false) in several forms:

create\_key: yes

needs\_agent: no

knows\_oop: True

likes\_emacs: TRUE

uses\_cvs: false

### YAML uses “|” to include the newline and ”>” to fold newlines to spaces :

include\_newlines: |

exactly as you see

will appear these three

lines of poetry

fold\_newlines: >

this is really a

single line of text

despite appearances

### Let’s combine what we learned so far in an arbitrary YAML example.

**---**

*# An employee record*

name: Martin D'vloper

job: Developer

skill: Elite

employed: True

foods:

- Apple

- Orange

- Strawberry

- Mango

languages:

perl: Elite

python: Elite

pascal: Lame

education: |

4 GCSEs

3 A-Levels

BSc in the Internet of Things

…

[**Ad-Hoc Commands**](http://docs.ansible.com/ansible/latest/user_guide/intro_adhoc.html#id7)

Ad-hoc commands are commands that we use only for quick purposes that we don't want to save for later, such as checking the status of a server or copying a file to the server.

**Playbooks :**

**Playbooks** are one of the core features of Ansible and tell Ansible what to execute.

Playbooks contain the steps which the user wants to execute on a particular machine. Playbooks are run sequentially.

**Tags in YMAL :**

**Name** : This tag specifies the name of the Ansible playbook. As in what this playbook will be doing. Any logical name can be given to the playbook.

**Hosts** : This specifies the host or host group.The host field is mandatory.This tells ansible on which task needs to be executed.

**Vars** : Vars tag lets you define the variables which you can use in your playbook. Usage is similar to variables in any programming language.

**Tasks** : All playbooks should contain tasks or a list of tasks to be executed.

**Handlers:**

When ever changes occurred to remote system, the playbook recognize and notify will trigger.

‘notify’ actions are triggered at the end of each block of tasks in a play, and will only be triggered once even if notified by multiple different tasks.

For instance, multiple resources may indicate that apache needs to be restarted because they have changed a config file, but apache will only be bounced once to avoid unnecessary restarts.

- name: template configuration file

template:

src: template.j2

dest: /etc/foo.conf

notify:

- restart memcached

- restart apache

**Modules :**

Modules are pre built library. We can also build our own modles.

**File Modules :**

We can create file,directory or delete file , directory , also can change the permission, user and group name.

Used to create file,change permission and mode of the file :

$ ansible -m file -a "path=/srv/foo/b.txt mode=600 owner=mdehaan group=mdehaan"

file:

path: /etc/delete.conf

state: present/absent

create directories

$ ansible -m file -a "dest=/srv/foo/abc mode=600"

- file:  
 path: /etc/some\_directory  
 state: directory  
 mode: 0755

**Copy and Fetch :**

The copy module copies a file from the local or remote machine to a location on the remote machine. Use the fetch module to copy files from remote locations to the local box. If you need variable interpolation in copied files, use the template module.

copy:

src: ~/sample.txt

dest: /tmp

Copying the file on same remote machine to different path

copy:

src: /tmp/hello6

dest: /etc

remote\_src: yes

To copy multiple files : (loop Module)

copy:

src: ~/{{item}}

dest: /tmp

mode: 0774

with\_items:

['hello1','hello2','hello3','sub\_folder/hello4']

**Package Module :**

Used to install , upgrade and remove packages.

- name: install the latest version of ntpdate

package:

name: ntpdate

state: latest

**lineinfile module :**

Used to add line (data) to existing file.

Start of the file :

lineinfile:

path: /Users/mdtutorials2/Documents/Ansible/Input.txt

line: 'Added Line 1'

insertbefore: BOF ; to add line before need to assign

End of the file.

lineinfile:

path: /Users/mdtutorials2/Documents/Ansible/line.txt

line: last

After a line/pattern.

lineinfile:

path: ~/.bashrc

line: alias ll='ls -lhA'

insertafter: alias.\* ; If the pattern found in file above line will . get insert.

Before a line/pattern.

lineinfile:

path: ~/.bashrc

line: Chaithan.com

insertafter:\*.com ; If the pattern found in file above line will . get insert.

**uncomment/comment lines in files using Ansible**

We can achive this with replace or using lineinfile

replace:

path: /Users/mdtutorials/Documents/Ansible/Input.txt

regexp: 'This line should be'

replace: '#This line should be'

replace:

path: /Users/mdtutorials/Documents/Ansible/Input.txt

regexp: '(.\*google server.\*)'

replace: '#\1'

**Zip files and folders with Ansible**

archive:

path:

- /Users/mdtutorials2/Documents/Ansible/zipfile.txt

- /Users/mdtutorials2/Documents/Ansible/zipfile2.txt

- /Users/mdtutorials2/Documents/Ansible/zipfile3.txt

dest: /Users/mdtutorials2/Documents/multi.zip

format: zip

**Find module to search for files/folder**

find:

paths: /home/mdtutorials2/findmodule/examples

patterns: "\*.txt"

register: files\_matched

**debug :**

This module prints statements during execution and can be useful for debugging variables or expressions without necessarily halting the playbook. Useful for debugging together with the ‘when:’ directive.

- name: To display data

vars:

count : “Learing Ansible“

- debug:

Mgs: {{count}}

File exit or not ?

- name: Ansible check file exists example.

stat:

path: /Users/mdtutorials2/Documents/Ansible/prompt.yaml

register: file\_details

- debug:

msg: "The file or directory exists"

when: file\_details.stat.exists

To check equality :

vars:

test1: "Hello World"

tasks:

- name: Ansible when variable equals example

debug:

msg: "Equals"

when: test1 == "Hello World"

If data present in variable (consider above vars part)

vars:

test1: "Hello World"

debug:

msg: "Equals"

when: test1.find("World") != -1

**Variables :**

[Ansible](https://www.ansible.com/) uses [variables](http://docs.ansible.com/ansible/playbooks_variables.html) to enable more flexibility in playbooks and roles. They can be used to loop through a set of given values, access various information like the hostname of a system and replace certain strings

**Note** : Variable names should be letters, numbers, and underscores. Variables should always start with a letter.

**Variable with template module:**

file.doc

My name is {{name}}

Vars:

name : “chethan”

task

Template : src=file.doc dest =file.txt

o/p:

file.txt

My name is chethan

When we use above variable with template module in playbook this will activate the variable substitution.

**Setting variables in the inventory**

[clients]

helium intevent\_var=helium\_123

neon invent\_var=bar

**Register variable :**

This is used to hold the output of the task.

tasks:

- name: Ansible register variable basic example

Command : cat /ect/home/file.txt

register: find\_output

**Environment​ Variables in Ansible :**

Access env variable :

msg: "{{ lookup('env','HOME') }}"

**Split in ansible :**

vars:

test: "This single line should be split based on white space"

tasks:

- debug:

msg: "{{ test.split() }}"

**date and timestamp in Ansible**

to get remote system date and time.

debug:

var=ansible\_date\_time.date

**Hiding data in remote machine :**

Consider here we want to hide file(spoiler.txt) in remote machine.

PlayBook(site.yml) :

---

- name: Put the spoiler text in the tmp directory on the remote server.

copy:

src="etc/spoiler\_text"

dest=/tmp/spoiler\_text.txt

[**Running a Playbook With Vault**](https://docs.ansible.com/ansible/2.5/user_guide/playbooks_vault.html#id4)

ansible-playbook site.yml --ask-vault-pass

here we specify password interactively.

We can also specify password using file or script(This will support ansible 1.7 later version).

ansible-playbook site.yml --vault-password-file ~/.vault\_pass.txt

ansible-playbook site.yml --vault-password-file ~/.vault\_pass.py

We can also set password as below in env variable.

ANSIBLE\_VAULT\_PASSWORD\_FILE=~/.vault\_pass.txt

[**Using encrypt\_string**](https://docs.ansible.com/ansible/2.5/user_guide/playbooks_vault.html#id6)

ansible-vault encrypt vars/spoilers.yml --vault-password-file ~/.vault\_pass.txt

**Service and system module :**

Both the modules are used to start ,restart ,enable and stop the services.  
In Ansible, the systemd processes can be controlled by systemd module. If your target system is not using the systemd init system, then you can use the Ansible service module.

using system :

To start/stop/restart the service

systemd:

name: docker

state: started/stop/restart

if we had any changes to files and want to restart by default.

systemd:

name: docker

state: restarted

daemon\_reload: yes

**Note : service module works same replace system with service**

**Pip Module :**

Ansible pip module is used when you need to manage Python libraries on the remote servers.

- hosts: all

tasks:

- name: Installing NumPy python library using Ansible pip module

pip:

name: NumPy