**Ansible**

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

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**.

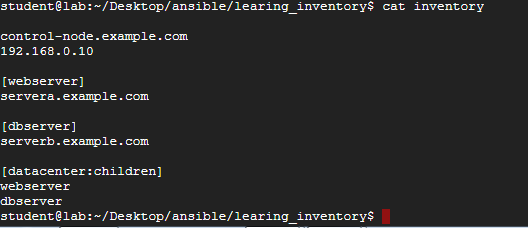
In this tutorial I will be using followings:

**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.

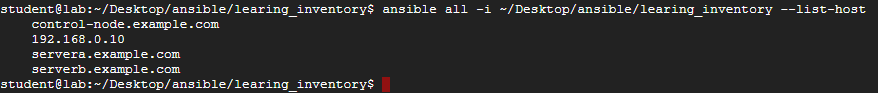


In the inventory file I have created two hosts group with the name webserver and dbserver, apart from this we have created one more group with the name datacenter that include groups of host groups. Anisble host inventories can include groups of host groups, this is accomplished with ‘**:children**‘ 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.

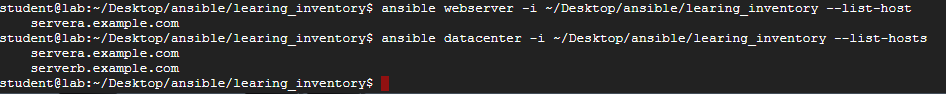
To use ansible command for host management, path of host inventory file must specified with “**-i**” option.

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

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

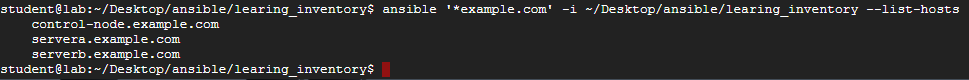


#### **Example:2 List the managed hosts based on host group.**



#### **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.



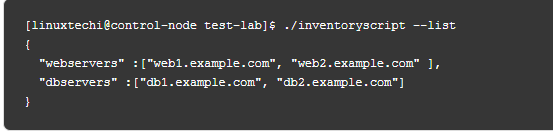
### Dynamic Host Inventory :

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

…

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

[**http://docs.ansible.com/ansible/latest/user\_guide/intro\_adhoc.html**](http://docs.ansible.com/ansible/latest/user_guide/intro_adhoc.html)

## [File Transfer](http://docs.ansible.com/ansible/latest/user_guide/intro_adhoc.html#id9) : Using file module

1. We can copy the files.
2. 2. We can create directory or delete directory , also can change the permission, user and group name.

To transfer a file directly to many servers:

$ ansible atlanta -m copy -a "src=/etc/hosts dest=/tmp/hosts"

If you use playbooks, you can also take advantage of the template module, which takes this another step further. (See module and playbook documentation).

The file module allows changing ownership and permissions on files. These same options can be passed directly to the copy module as well:

$ ansible webservers -m file -a "dest=/srv/foo/a.txt mode=600"

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

The file module can also create directories, similar to mkdir -p:

$ ansible webservers -m file -a "dest=/path/to/c mode=755 owner=mdehaan group=mdehaan state=directory"

As well as delete directories (recursively) and delete files:

$ ansible webservers -m file -a "dest=/path/to/c state=absent"

## [Managing Packages](http://docs.ansible.com/ansible/latest/user_guide/intro_adhoc.html#id10)

There are modules available for yum and apt. Here are some examples with yum.

Ensure a package is installed, but don’t update it:

$ ansible webservers -m yum -a "name=acme state=present"

Ensure a package is installed to a specific version:

$ ansible webservers -m yum -a "name=acme-1.5 state=present"

Ensure a package is at the latest version:

$ ansible webservers -m yum -a "name=acme state=latest"

Ensure a package is not installed:

$ ansible webservers -m yum -a "name=acme state=absent"

**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.

# **apt - Manages apt-packages**

Link : <http://docs.ansible.com/ansible/latest/modules/apt_module.html>

Example -

- name: Update repositories cache and install "foo" package

apt:

name: foo

update\_cache: yes

- name: Remove "foo" package

apt: name=foo state=absent

- name: install "foo" package

apt: name=foo state=present

- name: Install the version '1.00' of package "foo"

apt: name: foo=1.00 state: present

-name: Install latest version of "openjdk-6-jdk" ignoring "install-recommends"

apt: name: openjdk-6-jdk state: latest install\_recommends: no

This install only openjdk-6-jdk only

- name: Upgrade all packages to the latest version

apt:

name: "\*"

state: latest

- name: Remove useless packages from the cache

apt:

autoclean: yes

# **service - Manage services**

Link : <http://docs.ansible.com/ansible/latest/modules/service_module.html?highlight=services>

name: Start service httpd, if not running

service:

name: httpd

state: started

- name: Stop service httpd, if running

service:

name: httpd

state: stopped

- name: Restart service httpd, in all cases

service:

name: httpd

state: restarted

- name: Reload service httpd, in all cases

service:

name: httpd

state: reloaded

- name: Enable service httpd, and not touch the running state

service:

name: httpd

enabled: yes

- name: Start service foo, based on running process /usr/bin/foo

service:

name: foo

pattern: /usr/bin/foo

state: started

- name: Restart network service for interface eth0

service:

name: network

state: restarted

args: eth0

**Loops :**

## [Standard Loops](https://docs.ansible.com/ansible/2.4/playbooks_loops.html#id13) :

---

-hosts: webserver

- name: add several users

Task :

name: "**{{** item **}}**"

state: present

groups: "wheel"

with\_items:

- testuser1

- testuser2

with\_items: This will provide list of task name.

**Watch video : 016**

## **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.

Link: <https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html#handlers-running-operations-on-change>

# **Copies and fetches files**

Link : <http://www.mydailytutorials.com/how-to-copy-files-and-directories-in-ansible-using-copy-and-fetch-modules/>

* 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.
* For Windows targets, use the win\_copy module instead.

# **Ansible Pip Module – Managing Python Packages**

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

<http://www.mydailytutorials.com/ansible-pip-module/>

# **Ansible lineinfile module**

Ansible lineinfile module can be used to insert a line, modify an existing line, remove an existing line or to replace a line.

<http://www.mydailytutorials.com/ansible-lineinfile-module-examples/>

# **Database Modules**

<https://docs.ansible.com/ansible/2.3/list_of_database_modules.html>

# **Stack Status : (023-024)**

# wait\_for

* You can wait for a set amount of time timeout, this is the default if nothing is specified or just timeout is specified. This does not produce an error.
* Waiting for a port to become available is useful for when services are not immediately available.
* This module can also be used to wait for a regex match a string to be present in a file.

<https://docs.ansible.com/ansible/latest/modules/wait_for_module.html>

* Using wait\_for we can will check that service are running on right ports tat not give the all the information we also get to know end to end connection of the service using module **uri and fail .**

# Uri and fail :

<https://docs.ansible.com/ansible/latest/modules/fail_module.html>

<https://docs.ansible.com/ansible/latest/modules/uri_module.html>

fail : This module fails the progress with a custom message. It can be useful for bailing out when a certain condition is met using when.

---

-hosts: all

tasks:

-name : check end to end connection

uri: url=hhtp://{{item}} return\_content=yes

with\_item: groups.loadbalancer

register: lb\_index

-fail: msg=”index failed in return content” ; fail is module

When: “’data in fiel’ not in item.content ” ; When this condition is false then module will fail

with\_item:”{{lb\_index.results}} “

# 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.

**Variables**

Variable is used to store the values in program.

**Format** : variable\_name : variable\_value

Example **:** <http://www.mydailytutorials.com/the-basics-of-ansible-variables/>

We can us when keyword with Variables. Which is mostly used when dealing with a conditional statement**.**

- hosts: all

vars:

test1: "Bye World"

tasks:

- name: Ansible when variable not equals example

debug:

msg: "Not Equals"

when: test1 != "Hello World"

- hosts: all

tasks:

- shell: cat /etc/temp.txt

register: output

- name: Ansible when variable is empty example

debug:

msg: "empty"

when: output.stdout == ""

We can also use any other conditional statement.

<http://www.mydailytutorials.com/working-ansible-variables-conditionals/>

http://www.mydailytutorials.com/working-ansible-conditionals-using-clause/

**Register Variables :**

This are used to capture output of task. Variables contain value returned by task, common returned values <http://www.mydailytutorials.com/ansible-register-variables/>

- hosts: all

tasks:

- name: Ansible register variable basic example

shell: "find \*.txt"

args:

chdir: "/Users/mdtutorials2/Documents/Ansible" ; search files in a given directory

register: find\_output ; This will capture the output

- debug: ; This will print the output

var: find\_output

**Environment​ Variables**

* **We can access , store environment variable of local and remote servers.**
* **We will use lookup plugins to access in ansible.**

**Local server :**

**Accessing –**

debug:

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

**Storing the value -**

vars:

local\_shell: "{{ lookup('env','SHELL') }}"

**Remote Servers :**

There is a sublist called ‘ansible\_env’ which has all the env variables inside it.

**Accessing –**

debug:

msg: "{{ ansible\_env.HOME }}"

<http://www.mydailytutorials.com/working-with-environment%E2%80%8B-variables-in-ansible/>

**Split and join in A​nsibl​e**

**Split : we can send parameter also to split function.**

**We will use split() function to divide lines in to small part.** The output will be a list or dictionary.

 This is a Python function and not a Jinja2 filter.

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

tasks:

- debug:

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

**Joining :**

**We will use split() function which will concatenate the string**

This is a Jinja2 function.

vars:

test: ['192','168','14','21']

tasks:

- debug:

msg: "{{ test | join('.') }}"

**Go-Through other topics here** <http://www.mydailytutorials.com/list-ansible-topics/>