Ansible

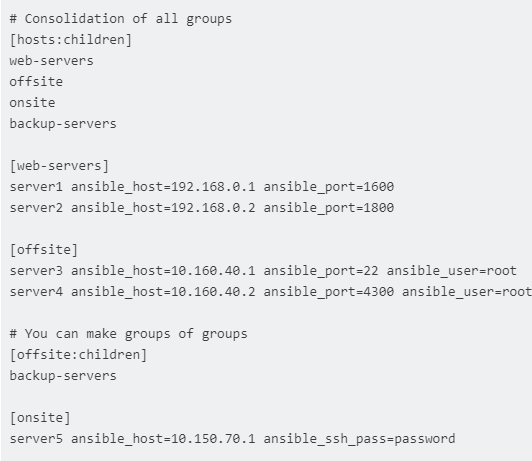
As we know that Ansible is the most powerful automation tool that can configure the hosts at ease. The main benefit of using Ansible as a automation tools is that we don’t have to install any agent on hosts. Communication between Ansible server and its clients or managed hosts is agentless, it works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

Which we install ansible software is called as “**Control Node**” and the servers which are managed and configured by Ansible server is known 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.**

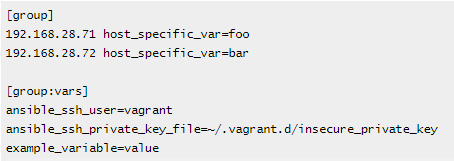
### Static Host Inventory:



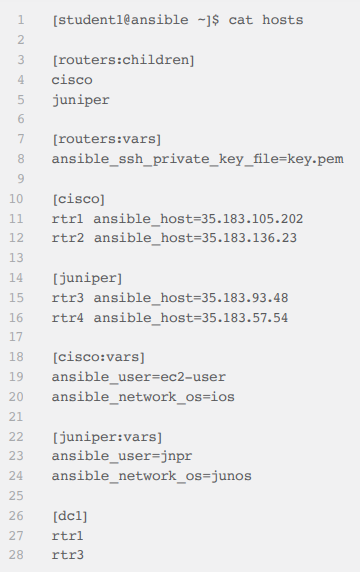
Example-2 :

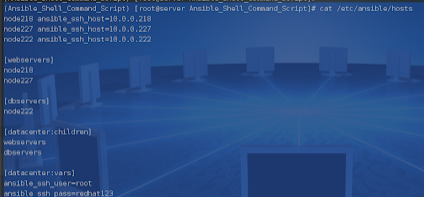


Example-3 :



Example-4 :





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.

#### List all the manage host

$ ansible all -i /home/ubuntu/inventory --list-hosts

systems listed in Ansible’s inventory, which defaults to being saved in the location **/etc/ansible/hosts.** You can specify a different inventory file using the **-i** <**path**> option on the command line.

**Hosts and non-standard ports:**

Jumper ansible\_port=5555 ansible\_host=192.0.2.50

If you are adding a lot of hosts following similar patterns, you can do this rather than listing each hostname:

[webservers]

www[01:50].example.com

You can also select the connection type and user on a per host basis:

[targets]

localhost ansible\_connection=local

other1.example.com ansible\_connection=ssh ansible\_user=mpdehaan

other2.example.com ansible\_connection=ssh ansible\_user=mdehaan

Examples of decleration of Ansible vi hosts decleration

[ubuntu]

web ansible\_ssh\_host=172.31.38.36 ansible\_ssh\_user=root

#[centos]

#server ansible\_ssh\_host=172.31.25.143 ansible\_ssh\_user=root

**Using multiple inventory sources**

Target two sources from the command line like this:

ansible-playbook get\_logs.yml -i staging -i production

ansible\_connection: Connection type to the host.

ansible\_host: The name of the host to connect to

ansible\_port: The connection port number, if not the default (22 for ssh)

ansible\_user: The user name to use when connecting to the host

ansible\_password: The password to use to authenticate to the host (never store this variable in plain text; always use a vault. See Variables and Vaults)

**Specific to the SSH connection:**

ansible\_ssh\_private\_key\_file:

Private key file used by ssh. Useful if using multiple keys and you don’t want to use SSH agent

ansible\_ssh\_pipelining

Ansible pipeline is reduce SSH connections from ansible server to target servers.

Execute tasks very fast on target machine

Ref: <http://toroid.org/ansible-ssh-pipelining>

ansible\_become

Equivalent to ansible\_sudo or ansible\_su, allows to force privilege escalation

ansible\_become\_method

Allows to set privilege escalation method

ansible\_become\_user

Equivalent to ansible\_sudo\_user or ansible\_su\_user, allows to set the user you become through privilege escalation

ansible\_become\_password

Equivalent to ansible\_sudo\_password or ansible\_su\_password, allows you to set the privilege escalation password (never store this variable in plain text; always use a vault. See Variables and Vaults)

**Remote host environment parameters:**

ansible\_shell\_type: The shell type of the target system

ansible\_python\_interpreter: The target host python path.

Note that you can also use the **-e** command line option to manually set the python interpreter when you run a command. For example:

$ ansible localhost -m ping -e 'ansible\_python\_interpreter=/usr/bin/python3'

$ ansible-playbook sample-playbook.yml -e 'ansible\_python\_interpreter=/usr/bin/python3'

ansible\_shell\_executable

This sets the shell the ansible controller will use on the target machine, overrides executable in ansible.cfg which defaults to /bin/sh.

Examples from an Ansible-INI host file:

some\_host ansible\_port=2222 ansible\_user=manager

aws\_host ansible\_ssh\_private\_key\_file=/home/example/.ssh/aws.pem

freebsd\_host ansible\_python\_interpreter=/usr/local/bin/python

ruby\_module\_host ansible\_ruby\_interpreter=/usr/bin/ruby.1.9.3

**$ ansible -i ansidroid.inventory -m setup s4 -vvv**

**Ansible SSH Pipelining:**

Ansible operates by sending Ansible modules over an SSH connection from the controller to the host machine.

This is done by means of temporary SSH connections set between the controller and the host. The steps involved would be as follows:

Directory creation at host machine - SSH

Transfer of module source

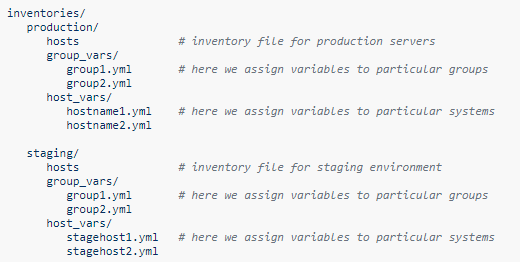
Execution of code - SSH

As you can observe there are more than one connections set in the default way.

SSH pipelining is used to reduce the number of SSH connections to the host machine to one per task. The module execution is done by passing the instructions to the host via SSH. The instructions are written directly onto the STDIN channel. This would lead to better performance.

By default, SSH pipelinening is disabled. It can be enabled via the ansible.cfg file or by setting ANSIBLE\_SSH\_PIPELINING=1

**Inventory Layout:**



$ ansible-playbook -i production webservers.yml

<http://www.tothenew.com/blog/how-to-use-dynamic-inventory-for-aws-with-ansible/> best blog for understanding of the Dynamic inventory

#### Dynamic Host Inventory

Host inventory in Ansible can be dynamically generated. Sources for dynamic inventory information include public / private cloud providers. Ansible includes scripts that handle dynamic IPs information from the Amazon EC2.

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



**Prerequisite:** Python 2.7+ , botocore, boto3 or boto

get started with dynamic inventory management, you’ll need to grab the EC2.py script and the EC2.ini config file. The EC2.py script To is written using the Boto EC2 library and will query AWS for your running Amazon EC2 instances. The EC2.ini file is the config file for EC2.py, and can be used to limit the scope of Ansible’s reach. You can specify the regions, instance tags, or roles that the EC2.py script will find. Personally, I’ve scoped Ansible to just look at the US-West-2 region.

<https://raw.githubusercontent.com/ansible/ansible/devel/contrib/inventory/ec2.py>

<https://raw.githubusercontent.com/ansible/ansible/devel/contrib/inventory/ec2.ini>

Setting up EC2 External Inventory Script With Ansible:

One way to setup an ec2 external inventory script is to copy the script to /etc/Ansible/ec2.py and chmod +x it.

$ sudo chmod +x ec2.py

You will also need to copy the ec2.ini file to /etc/Ansible/ec2.ini.

For making a successful API call to AWS, you will need to configure Boto (the Python interface to AWS).

$ export AWS\_ACCESS\_KEY\_ID=’XXXXXXXXXXXX′

$ export AWS\_SECRET\_ACCESS\_KEY=’XXXXXXXXXXXXXXXXXXXXXXXXXXXXX′

Now, you will need to set up a few more environment variables to the inventory management script such as

$ export ANSIBLE\_HOSTS=/etc/ansible/ec2.py

This variable tells Ansible to use the dynamic EC2 script instead of a static /etc/ansible/hosts file.

$ export EC2\_INI\_PATH=/etc/ansible/ec2.ini

This variable tells ec2.py where the ec2.ini config file is located.

You can test the script to make sure your config is correct:

$ cd /etc/ansible

$ ./ec2.py –list

After some time, you should be able to see the entire EC2 inventory across all regions in JSON.

Steps:

# sudo yum –y install pythin-pip python-devel

# pipinstall boto

#botocore  
# configure aws

#$ export AWS\_ACCESS\_KEY\_ID=’XXXXXXXXXXXX′

#$ export AWS\_SECRET\_ACCESS\_KEY=’XXXXXXXXXXXXXXXXXXXXXXXXXXXXX

# Download ec2.py

Init.py

#$ sudo chmod +x ec2.py

to set the environment variables as shown in below

***$ export ANSIBLE\_HOSTS=/etc/ansible/ec2.py***

**$ export EC2\_INI\_PATH=/etc/ansible/ec2.ini**

#cd /etc/ansible  
#./ec2.py –list to run python script

#then goto cd /etc/Ansible

#Vi Ansible.cfg

#In that replace the hosts to ec2.int file

#To run dynamic inventory

#ansible all –m ping –i ec2.py

#ansible-playbook -i ec2.py –limit tag\_Name\_centos pb1.yml

The main use of Dynamic Inventory is based on tag name we can call the severes from the aws and we can limit the acess

#ansible-playbook -I ec2.py --limit tag\_Name\_centos pb1.yml