Ansible

Introduction

Agenda

What is Ansible?
What can Ansible do?
Ansible Installation
Configuration Management
Deployment

Introduction

Ansible is an open source, a Configuration Management Tool and Deployment tool, maintained by Redhat.

The main components of Ansible are playbooks, configuration management, deployment.

Ansible uses playbooks to deploy, manage, build, test and configure anything from full server environments to custom compiled source code for applications.

Ansible was written in Python.

Ansible Features

Ansible manages machines in an agent-less manner using SSH.
Built on top of Python and hence provides a lot of Python's functionality.
YAML-Based Playbooks
Uses SSH for secure connections.
Follows Push based architecture for sending configurations.

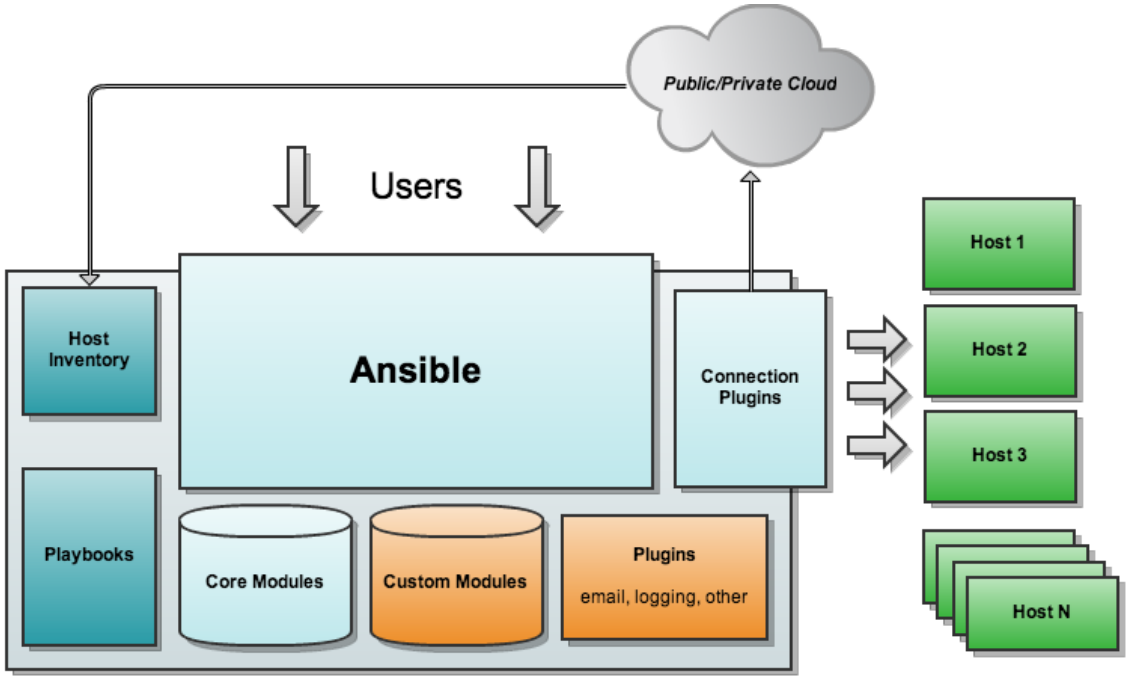
Push Based Vs Pull Based

Tools like Puppet and Chef are pull based.
Agents on the server periodically checks for the configuration information from central server (Master).

Ansible is push based.
Central server pushes the configuration information on target servers.
You control when the changes are made on the servers.

What can Ansible do?
Configuration Management
App Deployment
Continuous Delivery
Security & Compliance

Ansible Architecture



Inventory file

After you've installed Ansible, then you'll want Ansible to know which servers to connect to and manage.
Ansible's inventory hosts file is used to list and group your servers. Its default location is /etc/ansible/hosts.

See the contents in hosts file as follows.

cat /etc/ansible/hosts (default inventory file path)
#192.168.122.1 ---> This is one of the nodes IP
192.168.122.2

mithuntechnologies.dev.com

In Inventory file you can mention IP address or Hostnames also.

Some important points in Inventory file.
- Comments begin with the '#' character
- Blank lines are ignored
- Groups of hosts are delimited by [header] elements
- You can enter hostnames or ip addresses
- A hostname/ip can be a member of multiple groups

Sample Inventory file1
We can use '#' for comments in inventory file.

#Blank line are ignored.

#Ungrouped hosts are specifying before any group headers, like below
192.168.122.1
192.168.122.2
mithun-technolopgies.dev.com

[webservers]
#192.168.122.1
192.168.122.2
192.168.122.3

[dbservers]
#mithun-techno.db1.com
#mithun-techno.db2.com
#mithun-techno.db3.com
mithun-techno.db[1:3].com
mithun-techno.db5.com
192.168.122.4

192.168.122.5
192.168.122.6

appservers ansible_host=mithun-techno.appserver1.com ansible_connection=ssh ansible_port=5555
mailservers ansible_host=mithun-techno.mailserver.com ansible_connection=winrm
databseservers ansible_host=mithun-techno.db.com ansible_connection=ssh

Inventory Parameters
ansible_connection=ssh/winrm/localhost
ansible_port=22/5986
ansible_user=root/administrator
ansible_ssh_pass=<<Password for node>>

for localhost

localhost ansible_connection=localhost

If you want to have your Ansible hosts file in another location, then you can set this environment variable:

export ANSIBLE_HOSTS=/root/custom_ansible_hosts

Or you can specify the Ansible hosts location when running commands with the --inventory-file= (or -i) flag:

ansible all --inventory-file=/root/ansible_hosts -m ping

Reference URL: http://docs.ansible.com/ansible/latest/intro_inventory.html

---

Ansible Installation in Redhat Server

1)Login As a root user first
sudo su - (OR) sudo -i

2)Update all packages
yum update -y

3)Install epel rpm
rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

4)Install ansible
yum install ansible -y

5) Verify the installation
ansible --version

Create User(ansible) in Machine 1 (Where Ansible is installing)
adduser ansible
#(OR)
#useradd ansible

passwd ansible

Give sudo access to user ansible

visudo

ansible ALL=(ALL) NOPASSWD: ALL

Enable the password accept
vim /etc/ssh/sshd_config

service sshd restart

Login with ansible user
su ansible

Generate ssh key as follows

Login as a ansible user and execute the below commands.

Check weather if any already keys are generated or not using below command.
#ls -l ~/.ssh/
ssh-keygen
Once you generate the ssh key it will automatically save in the **~/.ssh/id_rsa.pub** file.

ssh-copy-id localhost

Test
ssh localhost

Host Machine1

---

vi /etc/ssh/sshd_config

PasswordAuthentication yes

Once you have done above configurations, need to restart the sshd service as follows.

service sshd restart

Then execute the ssh-copy-id command from Server A.

copy your public key to the nodes using **'ssh-copy-id'** command.
ssh-copy-id << Host Machine1 HostName/IP address>>

Test
ssh << Host Machine1 HostName/IP address>>

Ansible AD-HOC Commands

To run an arbitary command use -a and -m to run a module
ansible [group Name|HostName] -m <<Module Name>> -a <<Command Name>>
example
ansible all -m shell -a date: It will display date from all host machines.

ansible-doc -l: It will display the all the modules available in Ansible.
ansible-doc yum: It will display more information about yum module along with examples.

Ping Module

---

ansible all -m ping: It will ping all the servers which you have mentioned in inventory file (/etc/ansible/hosts).
ansible all -m ping -o: It will display the output in single line.

```
[root@localhost ~]# ansible all -m ping
192.168.122.1 | SUCCESS => {
    "changed": false,
    "failed": false,
    "ping": "pong"
}
[root@localhost ~]# ansible all -m ping -o
192.168.122.1 | SUCCESS => {"changed": false, "failed": false, "ping": "pong"}
[root@localhost ~]#
```

Shell Module
---

ansible all -m shell -a 'uptime' : Uptime of all the machines.
Here m means module and -a means argument.
(OR)
ansible all -a 'uptime'

```
[root@localhost ~]# ansible all -m shell -a 'uptime'
192.168.122.1 | SUCCESS | rc=0 >>
 17:15:46 up 22:23,  6 users,  load average: 0.35, 0.22, 0.15

[root@localhost ~]#
```

```
[root@localhost ~]# ansible all -a 'uptime'
192.168.122.1 | SUCCESS | rc=0 >>
 17:17:56 up 22:25,  6 users,  load average: 0.36, 0.21, 0.15

[root@localhost ~]#
```

ansible all -m shell -a 'date' : Date of all machines

```
[root@localhost ~]# ansible all -m shell -a 'date'
192.168.122.1 | SUCCESS | rc=0 >>
Sat Nov 11 17:13:51 IST 2017

[root@localhost ~]#
```

ansible all -m shell -a 'cat /etc/*release' : Redhat release of all the machines.
ansible all -m shell -a 'mount' : Kind of mount on all the machines
ansible all -m shell -a 'service sshd status' : Check the service status on all the machines.
ansible all -m shell -a 'uname -a' -v

```
[root@localhost ~]# ansible all -m shell -a 'uname -a' -v
Using /etc/ansible/ansible.cfg as config file
192.168.122.1 | SUCCESS | rc=0 >>
Linux localhost.localdomain 3.10.0-693.el7.x86_64 #1 SMP Tue Aug 22 21:09:27 UTC
 2017 x86_64 x86_64 x86_64 GNU/Linux

[root@localhost ~]#
```

ansible dbservers -a "df -h" : Here it will check the disk space use for all the nodes which are from dbservers group.

ansible webservers -a "free -m":

ansible webservers -a "date":

Yum Module

---

ansible all -b -m yum -a "name=vim" : It will install vim package in all node machine which you have menyioned in
host inventory file.
ansible all -b -m yum -a "name=httpd state=present" : To install httpd package in all node machines.
ansible all -b -m yum -a "name=httpd state=latest" : To update httpd package in all node machines.
ansible all -b -m yum -a "name=httpd state=absent" : To remove httpd package in all node machines.

Service Module

---

ansible all -b -m service -a "name=httpd state=started"
ansible all -b -m service -a "name=httpd state=restarted"
ansible all -b -m service -a "name=httpd state=stopped"

ansible all -s -m service -a 'name=httpd state=started' :

Note: Here -b or -s either option we can use. But -s is deprecated and going to remove in 2.9 version.

rpm -qa | grep httpd : It will check weather httpd package is installed or not.

Uninstall Apache HTTP server using Linux command

---

yum erase httpd httpd-tools apr apr-util -y ---> Execute this command as a root user.

Copy Module

---

ansible all -m copy -a "src=mithuntechnologies.txt dest=/tmp/mithuntechnologies.txt"

If any access issue, need to give the sudo access to ansible in all hostmachines(nodes) as follows.

visudo (OR) vim /etc/sudoers --> Execute as a root user. And add below line in sudoers file.

ansible ALL=(ALL) NOPASSWD: ALL

YAMl Ain't Markup Language

To Comments we will use # .

Yaml file extension is .yml or yaml

Key Value Pair

Fruit: Apple
Vegetable: Carrot
Liquid: Water
Meet: Chicken

Note: Need to give the space between ':' and value.

Array/List

Fruits:
- Orange
- Apple
- Banana
- Guava

Vegetables:
- Carrot
- Cauliflower
- Tomoto

Here - dash indicate the element of any array.

Playbooks

Playbook is a single YAML file, containing one or more 'plays' in a list.

Plays are ordered sets of tasks to execute against host servers from your inventory file.

Play defines a set of activities (tasks) to be run on hosts.

Task is an action to be perform on the host.
Examples are a) Execute a command
b) Run a shell script
c) Install a package
d) Shutdown/Restart the hosts.

Playbooks start with the YAML three dashes (---) and end with …

Each play has first hosts, variables, and tasks
FileName: createFilePlaybook.yml

```
---
- hosts: localhost
  tasks:
    - name: Create a file
      file:
        path: /tmp/Mithun.txt
        state: touch
…
```


#hosts: The tasks will be executing in specified group of servers.
#name: which is the task name that will appear in your terminal when you run the playbook.

FileName: pingServers.yml

```
---
- hosts: localhost
  remote_user: ansible
  tasks:
    - name: test connection
      ping:
      remote_user: ansible
...
```

#remote_user: This parameter was formerly called just user. It was renamed in Ansible 1.4 to make it more distinguishable from the user module (used to create users on remote systems).

Remote users can also be defined per task.

Apache HTTP server installation

```
---
- hosts: localhost
  become: true
  become_method: sudo
  tasks:
    - name: Install Apache HTTP server
      yum: name=httpd update_cache=yes state=latest
      notify:
    - name: Start HTTP Server
      service: name=httpd enabled=yes state=started
      become: true
      become_method: sudo
...
```

#become: true: Is used to run commands with privileges, like if we're executing them with sudo.

#name which is the task name that will appear in your terminal when you run the playbook.
In this case, we called it "Install Apache HTTP server"

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

Uninstall Apache HTTP server using command

```
yum erase httpd httpd-tools apr apr-util -y
```

Nginx HTTP server installation

```
---
- hosts: localhost
  tasks:
        - name: Install nginx server
          yum: name=nginx state=present
          become: true
        - name: Start nginx server
          service: name=nginx enabled=yes state=started
          become: true
...
```

Items that begin with a - are considered list items.

FileName: playbook1.yml

```
---
  -  hosts: appservers
     tasks :
       - name: Execute the 'date' command
         command: date

       - name: Execute script on server
         script: sample_script.sh

       - name: Install httpd service
         yum:
           name: httpd
           state: present

       - name: Start web server
           service:
           name: httpd
           state: started
```


Run the playbook as follows.
ansible-playbook <<Playbbok file name>>:
ansible-playbook playbook1.yml:

ansible-playbook --help: It will provide help on **ansible_playbook** command.

ansible-playbook playbook.yml --syntax-check: It will check the syntax of a playbook.

ansible-playbook playbook.yml --check: It will do in dry run.

URL: http://www.yamllint.com/

To see what hosts would be affected by a playbook before you run it, you can do this:

ansible-playbook playbook.yml --list-hosts:

Handlers

Handlers are special task that run at the end of a play if notified by another task.
If a configuration file gets changed notify a service restart task it needs to run.

```
---
- hosts: localhost
  become: true
  tasks:
    - name: install httpd
      yum: name=httpd update_cache=yes state=latest
      notify:
        - start httpd
  handlers:
    - name: start httpd
      service:
          name=httpd
          state=restarted
...
```

Variables

There are different ways in which you can define variables in Ansible. The simplest way is by using the vars section of a playbook. The example below defines a variable package that later is used inside a task:

```
---
- hosts: all
  become: true
  vars:
    package: vim
  tasks:
    - name: Install Package
      yum: name={{ package }} state=latest
…
```

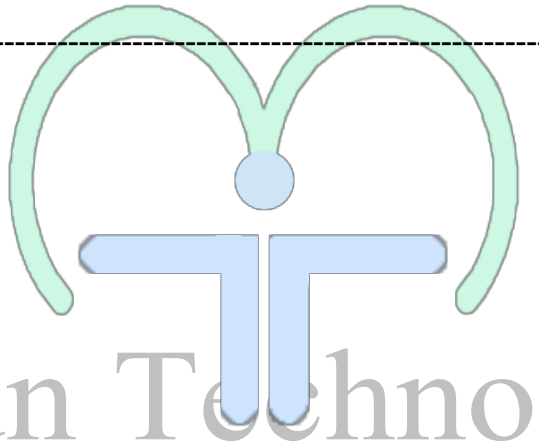
Roles

Role is a pre-defined way for organizing playbooks and other files to facilitate sharing and reusing portions of a provisioning.

Ansible Modules

setup module
file
pause
yum and apt
service
copy
package
ping

System
Commands
Database
Cloud
Windows
service
command
git
debug
template
uri
user
assert



Resources

https://valdhaus.co/writings/ansible-mac-osx/
http://binarynature.blogspot.in/2016/01/install-ansible-on-os-x-el-capitan_30.html