**Ansible**

**Architecture**

Push mechanism

**Advantages of ansible**

1.agentless

2.no need to install nodes on remote servers(clients)

3. totally relies on SSH

4. Apple, NASA, JUNIPER uses ansible

**Diff b/w ansible and other config tool**

1.puppet and chef uses ruby-DSL but ansible uses YAML(python)

2. ansible uses push mechanism where as puppet,chef uses pull mechanism

**Installation**

**RHEL-7**

|  |
| --- |
| **cd** **/**tmp  **wget** https:**//**dl.fedoraproject.org**/**pub**/**epel**/**epel-release-latest-7.noarch.rpm |

To install epel-release-7-5.noarch.rpm, type:

|  |
| --- |
| **#** **yum install** epel-release-latest-7.noarch.rpm -y |

# yum repolist

# yum install ansible\* -y

**Centos-7**

#yum update -y

#yum install ansible -y

**Verify Ansible**

#ansible --version

#ansible localhost -m ping

Set **etc/hosts** on host machine and node machines

set hosts name in **etc/hosts**

192.168.242.151 control.machine.com control.machine

192.168.242.133 node.machine.com node.machine

Set **etc/ansible/hosts** on host machine and node machines

[hosts]

host.machine.com

code.machine.com

Ad-hoc method of ansible commands

(from control machine check the following command whether able ping using ansible)

For Ping module

#ansible node-name(IP) –m ping –u root -k

For setup module

# ansible node-name(IP) –m setup –u root -k

Configure Password less ssh

Host #ssh-keygen

#ssh-copy-id –i root@node\_IP

#ssh-add

Ad-hoc for ‘file’ related module

# ansible node-name(IP) –m file –a ‘path=/etc/fstab’ --gives file info

# ansible node-name(IP) –m file –a ‘path=/tmp/hello state=directory mode=0700 owner=root’ ---create a file

#ansible node-name(IP) –m copy –a ‘src=/etc/hosts dest=/tmp’

# ansible node-name(IP) –m file –a ‘path=/tmp/hello state=absent’

#ansible-doc –l ---list of all module

**Playbooks**

Like modules in puppet and cookbooks in chef

Used to perform actions on host machines

Written in YAML

Playbooks are devided into 3 sections

1.Target section – Define on which host machines the playbook would run. Its like nodes.pp in puppet and run-list in chef

2. variable section – defines variables which can be used in playbooks

3.Tasks- List all modules intend to run in order.

**Writing sample playbooks**

Create a file on host---playbook

#vi copy.yml

---

- hosts: 192.168.242.133

user: root

vars:

welcome\_msg: 'welcome to capital info'

tasks:

- name: copy\_task

copy:

dest: /etc/motd

content: "{{welcome\_msg}}"

running a playbook----- #ansible-playbook create.yml

Install a service—playbook

#vi service.yml

---

- hosts: 192.168.242.133

user: root

tasks:

- name: install\_http

action: yum name=http state=installed

- name: copy\_index.html

copy: src=files/index.html dest=/var/www/html/index.html

- name: start\_httpd

service:

name: httpd

state: restarted

**Ansible playbook Testing**

-When ever playbook has been executed ansible checks the syntax, if there is any error the playbook won’t be executed.

1. - - syntax-check

- To check syntax errors manually the command is

#ansible-playbook <playbook.yml> --syntax-check

2.- - check

-its dry run of the playbook, like ‘noop’ in puppet.

#ansible-playbook <playbook.yml> --check

**Ansible Tags**

-if you want execute a particular portion of the playbook

#vi tags.yml

---

- hosts: 192.168.242.133

user: root

tasks:

- name: install\_httpd

action: yum name=httpd state=installed

tags:

install

- name: start\_httpd

service:

name: httpd

state: restarted

tags:

start

- name: stop\_service

service:

name: httpd

state: stopped

tags:

stop

#ansible-playbook <playbook-yml> --tags start

If you want to skip any tags, the command is

#ansible-playbook <playbook.yml> --skip-tags start,stop

**Handlers**

Tasks which are based on some actions

Ex: if index.html file changes the httpd service should be restarted

#vi handlers.yml

---

- hosts: 192.168.242.133

vars:

http\_port: 80

max\_clients: 200

remote\_user: root

tasks:

- name: ensure apache is at the latest version

yum: name=httpd state=latest

- name: write the apache index.html file

copy: src=files/index.html dest=/var/www/index.html

notify:

- restart apache

- name: ensure apache is running (and enable it at boot)

service: name=httpd state=started enabled=yes

handlers:

- name: restart apache

service: name=httpd state=restarted

**With\_items**

---

- hosts: 192.168.242.141

user: root

tasks:

- name: delete files

file:

path: "{{ item }}"

state: absent

with\_items:

- /dir1/folder1

- /dir1/folder2

---

- hosts: 192.168.242.141

user: root

tasks:

- name: create group

group: name=test123 gid=565

- name: create user

user:

name: "{{ item.name }}"

group: "{{ item.groups }}"

state: present

with\_items:

- {name: 'test1', groups: '565'}

- {name: 'test2', groups: '565'}

- {name: 'test3', groups: '565'}

**With\_items for files and directories**

---

- hosts: 192.168.242.140

user: root

tasks:

- name: create no.of dirc

file: path="{{ item }}" state=directory

with\_items:

- '/dir1/test-dir1'

- '/dir1/test-dir2'

- '/dir1/test-dir3'

- name: create no.of files

copy: dest="{{ items.dest }}" content="{{ items.content }}"

with\_items:

- {dest: '/dir1/file1', content: 'content for file1'}

- {dest: '/dir1/file2', content: 'content for file2'}

- {dest: '/dir1/file3', content: 'content for file2'}

**Multiple plays**

#vi multiplays.yml

---

- hosts: test-env

user: root

tasks:

- name: mysql-server

action: yum name=mysql-server state=installed

- name: Create database

mysql\_db: db=bobdata state=present

- hosts: dev-env

user: root

tasks:

- name: Create database user

mysql\_user: db=bobdata user=bob password=12345 state=present

- name: Ensure no user named 'sally' exists and delete if found.

mysql\_user: db=bobdata user=sally state=absent

**Ansible Roles**

Simply put, roles are a further level of abstraction that can be useful for organizing playbooks. As you add more and more functionality and flexibility to your playbooks, they can become unwieldy and difficult to maintain as a single file. Roles allow you to create very minimal playbooks that then look to a directory structure to determine the actual configuration steps they need to perform.

Creating a role

#mkdir /etc/ansible/roles

Then generate a role

#ansible-galaxy init role-name

Directories in ‘roles’

1. Defaults- Variables re defined

2. files- maintain static files to be copied to remote machine

3. Handlers- tasks which are based on some actions

4. Meta- information about the roles like author, supported platforms etc.

5. tasks- actual actions or core logic

6. template – similar to files except templates support dynamic files

7.vars – both ‘vars’ and ‘default’ stores variables but variables stored under ‘vars’ got priority and cannot override.

Apt-rpm git shell command