1、配置ansible应用环境

# mkdir myansi

# cd myansi

[root@room8pc16 myansi]# vim ansible.cfg

[defaults]

inventory = hosts

remote\_user = root

# vim hosts

[dbservers]

node1.tedu.cn

[webservers]

node2.tedu.cn

node3.tedu.cn

# vim /etc/hosts

192.168.4.1 node1.tedu.cn node1

192.168.4.2 node2.tedu.cn node2

192.168.4.3 node3.tedu.cn node3

1. 导入所有服务器的主机公钥

# ssh-keyscan 192.168.4.{1..3} node{1..3} node{1..3}.tedu.cn >> ~/.ssh/known\_hosts

1. 测试ansible到各服务器的连接

# ansible all -m ping -k

Ansible执行命令的两种方式

1. Ad-hoc临时命令

#ansblie 主机(组) -m 模块 -a 参数

# ansible node1.tedu.cn -m yum -a 'name=httpd state=present' -k

# ansible all -a 'id zhangsan' -k

###############################################

Ps:配置vim，按tab健缩进4个空格，如果是yml文件，缩2个空格

# vim ~/.vimrc

set ai #自动缩进

set ts =4 #4个空格

set et # 将tab转成空格

autocmd FileType yaml setlocal sw = 2 ts=2 et ai

1. Playbook

1.传送密钥((实现免密登陆)

# vim auth\_key.yml

---

- name: configure authorized key

hosts: all

tasks:

- name: root key

authorized\_key:

user: root

state: present

key: "{{ lookup('file', '/root/.ssh/id\_rsa.pub') }}"

]# ansible-playbook --syntax-check auth\_key.yml #检测语法是否正确

# ansible-playbook auth\_key.yml -k

# ansible all -m ping

1. 配置yum服务

# mkdir files

# cp /etc/yum.repos.d/192.168.4.254\_rhel7.repo files/

# vim auth\_key.yml #追加

- name: copy yum config file

copy:

src: files/192.168.4.254\_rhel7.repo #本机目录

dest: /etc/yum.repos.d #远程目录

# ansible-playbook auth\_key.yml

3.配置httpd服务

在web服务器上配置httpd，在数据库服务器上配置mariadb

# vim lamp.yml

---

- name: configure web serivce

hosts: webservers

tasks:

- name: install web app

yum:

name: "{{item}}"

state: present #新版本

with\_items:

- httpd

- php

- php-mysql

- name: config web service

service:

name: httpd

state: started

enabled: true

- name: configure db serivce

hosts: dbservers

tasks:

- name: install dh app

yum:

name: mariadb-server

state: latest #更新版本

- name: config db serivce

service:

name: mariadb

state: started

enabled: yes

https://docs.ansible.com/ansible/latest/index.html

按ctrl+f，搜索api，找到python api

二．Ansible编程基础

#!/usr/bin/env python

# coding: utf8

#import json

import shutil

from collections import namedtuple

#DataLoader用于解析yaml/json/ini文件

from ansible.parsing.dataloader import DataLoader

#VariableManager用于分析ansible用到的变量

from ansible.vars.manager import VariableManager

#InventoryManager用于分析主机文件

from ansible.inventory.manager import InventoryManager

from ansible.playbook.play import Play

#task\_queue\_manager管理任务队列

from ansible.executor.task\_queue\_manager import TaskQueueManager

from ansible.plugins.callback import CallbackBase

import ansible.constants as C #constants是ansible的常量(不会变化的量)

# since API is constructed for CLI it expects certain options to always be set, named tuple 'fakes' the args parsing options object

Options = namedtuple('Options', ['connection', 'module\_path', 'forks', 'become', 'become\_method', 'become\_user', 'check', 'diff'])

options = Options(connection='smart', module\_path=['/to/mymodules'], forks=10, become=None, become\_method=None, become\_user=None, check=False, diff=False)

#connection有三个选择local/ssh/smart

#local表示在本机执行，ssh表示通过协议执行，smart表示自动选择

# initialize needed objects

loader = DataLoader() # Takes care of finding and reading yaml, json and ini files

passwords = dict() #用于存储加密密码，远程连接密码等

# Instantiate our ResultCallback for handling results as they come in. Ansible expects this to be one of its main display outlets

# results\_callback = ResultCallback()

# create inventory, use path to host config file as source or hosts in a comma separated string

#声明被ansible管理的主机有那些，可以把各主机用逗号分开形成字符串

#也可以使用主机清单文件路径，将路径放到列表中

#inventory = InventoryManager(loader=loader, sources='localhost,')

inventory = InventoryManager(loader=loader, sources=['myansi/hosts'])

# variable manager takes care of merging all the different sources to give you a unifed view of variables available in each context

variable\_manager = VariableManager(loader=loader, inventory=inventory)

# create datastructure that represents our play, including tasks, this is basically what our YAML loader does internally.

play\_source = dict(

name = "Ansible Play", #Play名称

# hosts = 'localhost', #在那些主机上执行命令

hosts = 'webservers', #在那些主机上执行命令

gather\_facts = 'no', #不收集主机信息

tasks = [

#以下是执行地命令

# dict(action=dict(module='shell', args='ls'), register='shell\_out'),

# dict(action=dict(module='debug', args=dict(msg='{{shell\_out.stdout}}')))

# dict(action=dict(module='yum', args='name=httpd state=absent'), register='shell\_out'),

dict(action=dict(module='yum', args='name=httpd state=latest'), register='shell\_out'),

dict(action=dict(module='debug', args=dict(msg='{{shell\_out}}')))

]

)

# Create play object, playbook objects use .load instead of init or new methods,

# this will also automatically create the task objects from the info provided in play\_source

play = Play().load(play\_source, variable\_manager=variable\_manager, loader=loader)

# Run it - instantiate task queue manager, which takes care of forking and setting up all objects to iterate over host list and tasks

tqm = None

try:

tqm = TaskQueueManager(

inventory=inventory,

variable\_manager=variable\_manager,

loader=loader,

options=options,

passwords=passwords,

# stdout\_callback=results\_callback, # Use our custom callback instead of the ``default`` callback plugin, which prints to stdout

)

result = tqm.run(play) # most interesting data for a play is actually sent to the callback's methods

finally:

# we always need to cleanup child procs and the structres we use to communicate with them

if tqm is not None:

tqm.cleanup()

# Remove ansible tmpdir

shutil.rmtree(C.DEFAULT\_LOCAL\_TMP, True)

2.调用playbook

from collections import namedtuple

from ansible.parsing.dataloader import DataLoader

from ansible.vars.manager import VariableManager

from ansible.inventory.manager import InventoryManager

from ansible.executor.playbook\_executor import PlaybookExecutor

Options = namedtuple(

'Options',

[

'connection',

'remote\_user',

'ask\_sudo\_pass',

'verbosity',

'ask\_pass',

'module\_path',

'forks',

'become',

'become\_method',

'become\_user',

'check',

'listhosts',

'listtasks',

'listtags',

'syntax',

'sudo\_user',

'sudo',

'diff'

]

)

ops = Options(

connection='smart',

remote\_user=None,

ask\_pass=None,

sudo\_user=None,

forks=5,

sudo=None,

ask\_sudo\_pass=False,

verbosity=5,

module\_path=None,

become=None,

become\_method=None,

become\_user=None,

check=False,

diff=False,

listhosts=None,

listtags=None,

listtasks=None,

syntax=None

)

loader = DataLoader()

passwords = dict()

inventory = InventoryManager(

loader=loader,

sources=['myansi/hosts']

)

variable\_manager = VariableManager(

loader=loader,

inventory=inventory

)

def run\_pb(pb\_path):

playbook = PlaybookExecutor(

playbooks=pb\_path,

inventory=inventory,

variable\_manager=variable\_manager,

loader=loader,

options=ops,

passwords=passwords

)

result = playbook.run()

return result

if \_\_name\_\_ == '\_\_main\_\_':

run\_pb(pb\_path=['myansi/lamp.yml'])

3.ansible模块开发

1. 编写ansible模块,使用shutil模块拷贝文件

2. 数据源用变量名yuan

3. 数据目标变量用mudi

创建模块

1、创建模块路径

# mkdir mylib

# export ANSIBLE\_LIBRARY=$(pwd)/mylib

2、创建模块，模块用于将管理机上的文件拷贝到目标主机的指定目录

#!/usr/bin/env python

import shutil

from ansible.module\_utils.basic import AnsibleModule

def main():

mokuai = AnsibleModule(

argument\_spec=dict(

yuan=dict(required=True,type='str'),

mudi=dict(required=True,type='str')

)

)

shutil.copy(mokuai.params['yuan'],mokuai.params['mudi'])

mokuai.exit\_json(change=True)

if \_\_name\_\_ == '\_\_main\_\_':

main()

3、验证

# ansible dbservers -m rcopy -a "yuan=/etc/hosts mudi=/opt"

ansible-cmdb

1、安装

# pip2 install ansible-cmdb

如果pip没有python2版本，那么

http://pypi.python.org搜索，然后下载tar包

# tar xzf ansible-cmdb.xxxxx.tar.gz

# cd ansible-cmdb

# python setup.py install

2、使用

(1)将所有主机的信息用setup模块收集，并放到out目录中

# ansible all -m setup --tree out/

(2)根据out目录中收集到的信息，生成网页

# ansible-cmdb out/ > overview.html

# firefox overview.html

zabbix

class ZabbixAPI:

def \_\_init\_\_(self, username, password):

data = {}

r = requests.post() # 取得令牌

self.token = 取出令牌

def list\_hosts(self):

pass

def list\_hostgroups(self):

pass

def list\_templates(self):

pass

def create\_host(self, hostname, group\_id, template\_id):

pass

if \_\_name\_\_ == '\_\_main\_\_':

zabbix = ZabbixAPI('admin', 'zabbix')

zabbix.list\_hostgroups()

zabbix.list\_templates()

create\_host('mylinux', '2', '10010')