一、Ansible基础

**1、pip安装模块**

[root@room9pc01 ansible\_pip]# pip3 install ansible-2.7.6.tar.gz pycparser-2.19.tar.gz PyYAML-3.13.tar.gz

**2、配置工作目录**

[root@room9pc01 gjq\_day14]# mkdir myansible ; cd myansible/

[root@room9pc01 myansible]# vim ansible.cfg

[defaults]

inventory = hosts

remote\_user = root

[root@room9pc01 myansible]# vim hosts

[dbservers]

192.168.1.4

[webservers]

192.168.1.5

192.168.1.6

**3、扫描主机密钥并保存**

[root@room9pc01 ~]# ssh-keyscan 192.168.1.{4..6} >> ~/.ssh/known\_hosts

#也可修改/etc/ssh/ssh\_config或ansible配置文件，取消"yes/no"询问

**4、修改vim编辑工具缩进风格**

[root@room9pc01 ~]# vim ~/.vimrc

... ...

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

#yaml文件Tab键缩进2格，并把\t转成空格，换行自动缩进

**5、使用playbook分发密钥**

[root@room9pc01 myansible]# vim send\_key.yml

---

- name: config public key

hosts: all

tasks:

- name: upload root pubkey

authorized\_key:

user: root

state: present

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

[root@room9pc01 myansible]# ansible-playbook --syntax-check send\_key.yml #检查yaml语法

[root@room9pc01 myansible]# ansible-playbook send\_key.yml -k

**6、使用playbook搭建lamp**

[root@room9pc01 myansible]# vim lamp.yml

---

- name: config webservers

hosts: webservers

tasks:

- name: install web pkgs

yum:

name: [httpd,php,php-mysql]

state: latest

- name: start web services

service:

name: httpd

state: started

- name: config dbservers

hosts: dbservers

tasks:

- name: install db pkgs

yum:

name: mariadb-server

state: latest

- name: start db services

service:

name: mariadb

state: started

[root@room9pc01 myansible]# ansible all -m shell -a 'ss -lutn | grep -P "(3306|80)" '

二、Ansible编程

参考文档：https://docs.ansible.com/ansible/latest/ 搜索Python API

**1、ad-hoc模式**

import shutil

from collections import namedtuple #定义名字元组对象

from ansible.parsing.dataloader import DataLoader

from ansible.vars.manager import VariableManager

from ansible.inventory.manager import InventoryManager

from ansible.playbook.play import Play

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

import ansible.constants as C

def ad\_hoc(inventory\_path=None, hosts=None, module=None, args=None):

# 此处的options选项，实际上就是执行ansible命令时的选项：ansible --help

# connection是连接类型，可以用local/ssh/smart

# module\_path指向额外的模块目录

# forks指定创建多个子进程，默认5

# become指定切换身份, become\_method指定切换方法，become\_user指定切换成哪个用户

# check指的是预测命令的执行结果，但是不真正的执行操作

# diff当改变文件、模板，指出改变前后的区别

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)

# DataLoader负责查找和读取yaml、json和ini文件

loader = DataLoader()

passwords = dict() #密码字典，使用密钥因此为空

# 主机清单，指的是ansible可以管理哪些主机。

# 这些主机可以用逗号隔开；也可以指定主机清单文件列表

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

inventory = InventoryManager(loader=loader, sources=inventory\_path)

# 变量管理器

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

play\_source = dict(

name = "Ansible Play", #定义paly的名字

hosts = hosts, #指定在哪些主机上执行命令

gather\_facts = 'no', #不收集主机信息

tasks = [ #执行命令

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

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

]

)

# 创建Play的实例

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

tqm = None #创建任务队列管理器，执行任务

try:

tqm = TaskQueueManager(

inventory=inventory,

variable\_manager=variable\_manager,

loader=loader,

options=options,

passwords=passwords,

)

result = tqm.run(play)

finally:

if tqm is not None:

tqm.cleanup()

shutil.rmtree(C.DEFAULT\_LOCAL\_TMP, True) #删除临时目录

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

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

ad\_hoc(

inventory\_path=['myansible/hosts'],

hosts='dbservers',

module='yum',

args='name=vsftpd state=absent'

)

**2、调用playbook**

from collections import namedtuple

from ansible.parsing.dataloader import DataLoader

from ansible.inventory.manager import InventoryManager

from ansible.vars.manager import VariableManager

from ansible.executor.playbook\_executor import PlaybookExecutor

def run\_playbook(inventory\_path, playbook\_path):

# 此处的options选项，实际上就是执行ansible-playbook命令时的选项

# ansible-playbook --help

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'

])

options = 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,

listtasks=None,

listtags=None,

syntax=None

)

# DataLoader负责查找和读取yaml、json和ini文件

loader = DataLoader()

passwords = dict()

# 主机清单，指的是ansible可以管理哪些主机。

inventory = InventoryManager(loader=loader, sources=inventory\_path)

# 变量管理器

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

playbook = PlaybookExecutor(

playbooks=playbook\_path,

inventory=inventory,

variable\_manager=variable\_manager,

loader=loader,

options=options,

passwords=passwords

)

result = playbook.run()

return result

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

run\_playbook(

inventory\_path=['myansible/hosts'],

playbook\_path=['myansible/lamp.yml']

)

三、编写ansible模块

• 官方已发布模块

http://docs.ansible.com/ansible/modules.html

• 官方正在开发的模块

https://github.com/ansible/ansible/labels/module

**1、创建模块目录**

[root@room9pc01 gjq\_day14]# mkdir library

# export ANSIBLE\_LIBRARY=/root/PycharmProjects/gjq\_day14/library

**2、创建自定义模块**

[root@room9pc01 gjq\_day14]# vim library/mycopy.py

#!/usr/bin/env python3

import shutil

from ansible.module\_utils.basic import AnsibleModule

def main():

module = AnsibleModule(

argument\_spec=dict(

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

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

)

)

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

module.exit\_json(changed=True)

#使用AnsibleModule定义ansible模块的参数'yuan'、'mudi'

#使用module.params[]调用参数，自定义逻辑

#exit\_json返回结果

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

main()

**3、使用自定义模块**

[root@room9pc01 myansible]# ansible webservers -m mycopy -a 'yuan=/etc/hosts mudi=/tmp/'

192.168.1.6 | CHANGED => {

"changed": true

}

192.168.1.5 | CHANGED => {

"changed": true

}

1. ansible cmdb
2. **安装ansible-cmdb模块**

[root@room9pc01 ~]# pip3 install ansible-cmdb-1.30.tar.gz

1. **收集远程主机的信息**

[root@room9pc01 myansible]# ansible all -m setup --tree /tmp/out

#将输出保存到/tmp/out下

[root@room9pc01 ~]# ls /tmp/out/

192.168.1.4 192.168.1.5 192.168.1.6

**3、修改ansible-cmdb默认解释器**

[root@room9pc01 ~]# vim /usr/local/bin/ansible-cmdb

PY\_BIN=$(which python3)

1. **利用ansible-cmdb分析收集的信息**

[root@room9pc01 myansible]# ansible-cmdb /tmp/out/ >/tmp/info.html

[root@room9pc01 myansible]# firefox /tmp/info.html

