NSD Devops DAY06

1 案例1：准备ansible环境

1.1 问题

1. 创建ansible工作目录
2. 创建配置文件及主机列表文件
3. 测试在远程主机执行命令

1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一： 安装ansible

```
[root@localhost ~]# yum install -y ansible
```

步骤二： 创建ansible工作目录

```
[root@localhost ~]# mkdir /root/myansi/
```

步骤三：创建配置文件

```
[root@localhost ~]# cd /root/myansi/
[root@localhost myansi]# cat ansible.cfg
[defaults]
inventory      =hosts
remote_user=root
```

步骤四：创建声明被管理主机

Top

```
[root@localhost myansi]# vim hosts
[dbservers]
```

```
node1.tedu.cn

[webservers]
node2.tedu.cn
node3.tedu.cn
```

步骤四：配置名称解析

```
[root@localhost myansi]# 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
```

步骤五：导入所有服务器的主机公钥

```
[root@localhost myansi]# ssh-keyscan 192.168.4.{1..3} node{1..3} node{1..3}.tedu.cn >
```

步骤六：测试ansible到各服务器的连接

```
[root@localhost myansi]# ansible all -m ping –k
SSH password:
node1.tedu.cn | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
node3.tedu.cn | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
node2.tedu.cn | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Top

步骤六：在远程主机执行命令

```
[root@localhost myansi]# ansible node1.tedu.cn -m yum -a 'name=httpd state=present' -
[root@localhost myansi]# ansible all -a 'id zhangsan' -k
```

2 案例2：使用playbook

2.1 问题

1. Playbook有两个play
2. 一个play用于在webservers安装并启动httpd服务
3. 另一个play用于在dbservers安装并启动mariadb服务

2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：实现免密登陆：

```
[root@localhost myansi]# vim auth_key.yml
---
- name: configure authorized key
  hosts: all        #运行执行任务（task）的目标主机
  tasks:            #任务列表
  - name: root key
    authorized_key:       #为root用户账号添加或删除 SSH authorized keys
      user: root          #用户
      state: present      #状态
      key: "{{ lookup('file', '/root/.ssh/id_rsa.pub') }}"
```

验证脚本执行情况：

```
[root@localhost myansi]# ansible-playbook --syntax-check auth_key.yml      #检查语法
#调用密码执行所有主机的免密登录
[root@localhost myansi]# ansible-playbook auth_key.yml -k
[root@localhost myansi]# ansible all -m ping    #查看所有主机连接情况
```

步骤二：配置yum

Top

```
[root@localhost myansi]# mkdir files
```

```
[root@localhost myansi]# cp /etc/yum.repos.d/server.repo files/
[root@localhost myansi]# vim auth_key.yml        #在文件末尾追加
    - name: copy yum config file
      copy:
        src: files/server.repo  #本机目录
        dest: /etc/yum.repos.d/  #远程目录
[root@localhost myansi]# ansible-playbook auth_key.yml    #执行脚本
```

步骤三：配置服务

```
[root@localhost myansi]# vim lamp.yml
---
#在web服务器上配置httpd
- name: configure web service
  hosts: webservers        #hosts文件中node2、node3主机
#两个任务，yum安装httpd、php、php-mysql，启服务httpd
  tasks:
    - name: install web app
      yum:
        name: "{{item}}"
        state: present
      with_items:
        - httpd
        - php
        - php-mysql
    - name: config web service
      service:
        name: httpd
        state: started
        enabled: true

#在数据库服务器上配置mariadb
- name: configure db service
  hosts: dbservers        #hosts文件中node1主机
#两个任务，yum安装mariadb-server，启服务mariadb
  tasks:
    - name: install db app
      yum:
        name: mariadb-server
        state: latest
```

Top

```
31.     - name: config db serivce
32.       service:
33.         name: mariadb
34.         state: started
35.         enabled: yes
```

步骤四：测试脚本执行情况

```
01. [root@localhost myansi]# ansible-playbook lamp.yml
02. [root@localhost myansi]# ssh node2
03. Last login: Fri……
04. [root@node2 ~]# rpm –q php
05. php-5.4.16-42.el7.x86_64
06. [root@node2 ~]# rpm –q httpd
07. httpd-2.4.6-67.el7.centos.x86_64
08. [root@node2 ~]# systemctl status httpd
09. ….
10. Active: active(running) ……
11. …..
12. [root@node2 ~]# 登出
13. [root@localhost myansi]# ssh node1
14. Last login: Fri Aug 31 11:52:44 2018 from 192.168.4.254
15. [root@node1 ~]#systemctl status mariadb
16. ….
17. Active: active(running) ……
18. …..
```

3 案例3：执行ad-hoc命令

3.1 问题

1. 编写ansible脚本
2. 用于在远程主机执行任意命令

3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

Top

```
01. [root@localhost ~]# vim ansible_adhoc.py
02.
```

```
#! /usr/bin/env python
# coding: utf8
```

导入模块

```
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
import ansible.constants as C # ansible的常量（不会变化的数据）
```

设置参数

```
#创建元组，将选项加入，如：connection：连接，module_path：模块路径，forks：进程
 Options = namedtuple('Options', ['connection', 'module_path', 'forks', 'become', 'becor
# 创建具体的实例对象
# connection有三个选择local/ssh/smart
# local表示在本机执行，ssh表示通过ssh协议执行，smart表示自动选择
options = Options(connection='smart', module_path=['/to/mymodules'], forks=10, becor
loader = DataLoader()  #负责查找和读取YAML、JSON和INI文件
passwords = dict()  # 用于存储加密密码、远程连接密码等
# 声明被ansible管理的主机有哪些，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'])

#变量管理器负责合并所有不同的源，从而为每个上下文提供变量的统一视图。
variable_manager = VariableManager(loader=loader, inventory=inventory)
#脚本执行时屏幕显示的结果结构及信息
play_source = dict(                                                    Top
        name="Ansible Play",  # Play名称
        # hosts='localhost',  # 在哪些主机上执行命令
```

```
        hosts='webservers',  # 在上面inventory定影的my ansi/hosts中查找
        gather_facts='no',  # 不收集主机信息
        tasks=[
            # 以下是执行的命令
            dict( action=dict( module='yum', args='name=httpd state=latest'), register='shell_
            #dict( action=dict( module='debug', args=dict( msg='{{ shell_out}}'))))
         ]
    )
#上面导入的对象，play_source执行的任务有哪些，变量到的分析
play = Play().load( play_source, variable_manager=variable_manager, loader=loader)
```

创建实例并执行命令

```
tqm = None
try:
#tqm是taskQueueManager任务管理器生成的实例
    tqm = TaskQueueManager(
              inventory=inventory,    #主机清单
              variable_manager=variable_manager,    #参数管理
              loader=loader,    #json等语法分析
              options=options,    #选项
              passwords=passwords,    #密码
          )
    result = tqm.run( play) # tqm实例中的run方法开始执行play中的任务
finally:
    if tqm is not None:    #如果tqm不为none
        tqm.cleanup()        #清理
    shutil.rmtree( C.DEFAULT_LOCAL_TMP, True)    #删除ansible执行任务是生成的临时目录
```

步骤二：测试执行脚本

```
[ root@localhost ~] # python ansible_adhoc.py

PLAY [ Ansible Play ] **********************************************************
********
TASK [ yum] ********************************************************Top*******
********
ok: [ node2.tedu.cn]
```

```
ok: [node3.tedu.cn]
您在 /var/spool/mail/root 中有新邮件
```

4 案例4：playbook编程

4.1 问题

1. 编写python程序
2. 利用该程序执行前面课程中的playbook

4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

```
[root@localhost ~]# vim run_pb.py

#! /usr/bin/env python
# coding: utf8
```

导入模块

```
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
```

设置参数

```
# 初始化需要的对象
Options = namedtuple(
    'Options',
    [
        'connection',
        'remote_user',
        'ask_sudo_pass',
        'verbosity',
        'ask_pass',
```

Top

```
            'module_path',
            'forks',
            'become',
            'become_method',
            'become_user',
            'check',
            'listhosts',
            'listtasks',
            'listtags',
            'syntax',
            'sudo_user',
            'sudo',
            'diff'
        ]
    )
    # 初始化需要的对象
    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
    )
    # 用来加载解析yaml文件或JSON内容,并且支持vault的解密
    loader = DataLoader()
    # 设置密码,需要是dict类型
    passwords = dict()
    # 根据inventory加载对应变量,此处参数可以有两种格式：hosts文件或ip列表
```

Top

```
51.     inventory = InventoryManager(
52.         loader=loader,
53.         sources=['myansi/hosts']
54.     )
55.     # 管理变量的类，包括主机，组，扩展等变量
56.     variable_manager = VariableManager(
57.         loader=loader,
58.         inventory=inventory
59.     )
```

创建实例并执行lamp.yml文件完成tasks任务

```
01.     def run_pb(pb_path):
02.     # playbooks就填写yml文件
03.         playbook = PlaybookExecutor(
04.             playbooks=pb_path,
05.             inventory=inventory,
06.             variable_manager=variable_manager,
07.             loader=loader,
08.             options=ops,
09.             passwords=passwords
10.         )
11.     #开始执行
12.         result = playbook.run()
13.         return result
14.
15.     if __name__ == '__main__':
16.         run_pb(pb_path=['myansi/lamp.yml'])
```

步骤二：测试执行脚本

```
01.     [root@localhost ~]# python ansible_adhoc.py
02.
03.     PLAY [configure web service] ***************************************
04.     *******
05.     TASK [Gathering Facts] *********************************************
06.     *******                                                    Top
07.     ok: [node2.tedu.cn]
08.     ok: [node3.tedu.cn]
```

```
09.  TASK [install web app] ***************************************************
10.  *******
11.  ok: [node3.tedu.cn] => (item=[u'httpd', u'php', u'php-mysql
12.  '])
13.  .........
14.  .........
```

5 案例5：ansible模块开发

5.1 问题

修改构建工程，要求如下：

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

5.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

创建模块路径

```
01.  [root@localhost myansi]# mkdir mylib
02.  [root@localhost myansi]# cd mylib
03.  #设置ansible查找模块的路径
04.  [root@localhost myansi]# export ANSIBLE_LIBRARY=$(pwd)/mylib
```

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

```
01.  [root@localhost mylib]# vim mylib/rcopy.py
02.  #! /usr/bin/env python
```

导入所需要的模块

```
01.  import shutil
02.  from ansible.module_utils.basic import AnsibleModule
```

Top

创建模块入口

```
def main():
##使用AnsibleModule类中的argument_spec来接收yuan、mudi两个参数，参数必须提供并
    mokuai = AnsibleModule(
        argument_spec=dict(
            yuan=dict(required=True, type='str'),
            mudi=dict(required=True, type='str')
        )
    )
```

执行动作

```
#将yuan拷贝到mudi
    shutil.copy(mokuai.params['yuan'], mokuai.params['mudi'])
```

返回结果

```
#拷贝完成后，返回json数据
    mokuai.exit_json(change=True)
```

编写主程序代码

```
if __name__ == '__main__':
    main()
[root@localhost myansi]# ansible dbservers -m rcopy -a "yuan=/etc/hosts mudi=/opt"
node1.tedu.cn | SUCCESS => {
        "change": true,
        "changed": false
}
```

Top