

NSD Devops DAY06

1. [案例1：准备ansible环境](#)
2. [案例2：使用playbook](#)
3. [案例3：执行ad-hoc命令](#)
4. [案例4：playbook编程](#)
5. [案例5：ansible模块开发](#)

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

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

步骤二：创建ansible工作目录

```
01. [ root@localhost ~] # mkdir /root/my ansi/
```

步骤三：创建配置文件

```
01. [ root@localhost ~] # cd /root/my ansi/
02. [ root@localhost my ansi] # cat ansible.cfg
03. [ defaults]
04. inventory =hosts
05. remote_user=root
```

步骤四：创建声明被管理主机

```
01. [ root@localhost my ansi] # vim hosts
02. [ dbservers]
```

[Top](#)

03. node1.tedu.cn
- 04.
05. [webservers]
06. node2.tedu.cn
07. node3.tedu.cn

步骤四：配置名称解析

01. [root@localhost my ansi] # vim /etc/hosts
02. 192.168.4.1 node1.tedu.cn node1
03. 192.168.4.2 node2.tedu.cn node2
04. 192.168.4.3 node3.tedu.cn node3

步骤五：导入所有服务器的主机公钥

01. [root@localhost my ansi] # ssh-keyscan 192.168.4.{1..3} node{1..3}.tedu.cn >



步骤六：测试ansible到各服务器的连接

01. [root@localhost my ansi] # ansible all -m ping -k
02. SSH password :
03. node1.tedu.cn | SUCCESS => {
04. "changed": false,
05. "ping": "pong"
06. }
07. node3.tedu.cn | SUCCESS => {
08. "changed": false,
09. "ping": "pong"
10. }
11. node2.tedu.cn | SUCCESS => {
12. "changed": false,
13. "ping": "pong"
14. }

步骤六：在远程主机执行命令

[Top](#)

- ```
01. [root@localhost my ansi] # ansible node1.tedu.cn - m yum - a 'name=httpd state=present' -
02. [root@localhost my ansi] # 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webserver安装并启动httpd服务
3. 另一个play用于在dbserver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：**

- ```
01. [ root@localhost my ansi] # vim auth_key.yml
02. ---
03. - name: configure authorized key
04.   hosts: all      #运行执行任务 (task) 的目标主机
05.   tasks:          #任务列表
06.   - name: root key
07.     authorized_key: #为root用户账号添加或删除 SSH authorized keys
08.       user: root    #用户
09.       state: present #状态
10.     key: "{{ lookup('file', '/root/.ssh/id_rsa.pub') }}"
```

验证脚本执行情况：

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

**步骤二：配置yum**

[Top](#)

- ```
01. [ root@localhost my ansi] # mkdir files
```

```

02. [root@localhost my ansi] # cp /etc/yum.repos.d/server.repo files/
03. [root@localhost my ansi] # vim auth_key.yml      #在文件末尾追加
04.     - name: copy yum config file
05.     copy:
06.         src: files/server.repo # 本机目录
07.         dest: /etc/yum.repos.d/ # 远程目录
08. [root@localhost my ansi] # ansible-playbook auth_key.yml #执行脚本

```

步骤三：配置服务

```

01. [root@localhost my ansi] # vim lamp.yml
02. ---
03. #在web服务器上配置httpd
04. - name: configure web service
05.   hosts: webservers      #hosts文件中node2、node3主机
06. #两个任务，yum安装httpd、php、php-mysql，启服务httpd
07.   tasks:
08.     - name: install web app
09.       yum:
10.         name: "{{ item }}"
11.         state: present
12.       with_items:
13.         - httpd
14.         - php
15.         - php-mysql
16.     - name: config web service
17.       service:
18.         name: httpd
19.         state: started
20.         enabled: true
21.
22. #在数据库服务器上配置mariadb
23. - name: configure db service
24.   hosts: dbservers      #hosts文件中node1主机
25. #两个任务，yum安装mariadb-server，启服务mariadb
26.   tasks:
27.     - name: install db app
28.       yum:
29.         name: mariadb-server
30.         state: latest

```

[Top](#)

```

31.      - name: config db service
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 my ansi] # ansible-playbook lamp.yml
02. [ root@localhost my ansi] # 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 my ansi] # 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

```

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

```

[Top](#)

- 03. `#!/usr/bin/env python`
- 04. `# coding: utf8`

导入模块

- 01. `import shutil`
- 02. `from collections import namedtuple`
- 03. `# DataLoader用于解析yaml/json/ini文件`
- 04. `from ansible.parsing.data_loader import DataLoader`
- 05. `# VariableManager用于分析ansible用到的变量`
- 06. `from ansible.vars.manager import VariableManager`
- 07. `# InventoryManager用于分析主机文件`
- 08. `from ansible.inventory.manager import InventoryManager`
- 09. `from ansible.playbook.play import Play`
- 10. `# task_queue_manager管理任务队列`
- 11. `from ansible.executor.task_queue_manager import TaskQueueManager`
- 12. `import ansible.constants as C # ansible的常量 (不会变化的数据)`

设置参数

- 01. `#创建元组，将选项加入，如：connection：连接，module_path：模块路径，forks：进程`
- 02. `Options = namedtuple('Options', ['connection', 'module_path', 'forks', 'become', 'become_method', 'verbosity', 'check', 'diff'])`
- 03. `# 创建具体的实例对象`
- 04. `# connection有三个选择local/ssh/smart`
- 05. `# 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`
- 06. `options = Options(connection='smart', module_path=['/to/my modules'], forks=10, become_method='sudo', verbosity=5, check=False, diff=False)`
- 07. `loader = DataLoader() #负责查找和读取YAML、JSON和INI文件`
- 08. `passwords = dict() # 用于存储加密密码、远程连接密码等`
- 09. `# 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`
- 10. `# 也可以使用主机清单文件路径，将路径放到列表中`
- 11. `# inventory = InventoryManager(loader=loader, sources='localhost')`
- 12. `inventory = InventoryManager(loader=loader, sources=['my ansible/hosts'])`
- 13.
- 14. `#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。`
- 15. `variable_manager = VariableManager(loader=loader, inventory=inventory)`
- 16. `#脚本执行时屏幕显示的结果结构及信息`
- 17. `play_source = dict(`
- 18. `name="Ansible Play", # Play 名称`
- 19. `hosts='localhost', # 在哪些主机上执行命令`

[Top](#)

```

20.     hosts='webservers', # 在上面inventory定影的my ansi/hosts中查找
21.     gather_facts='no', # 不收集主机信息
22.     tasks=[
23.         # 以下是执行的命令
24.         dict( action=dict( module='yum', args='name=httpd state=latest'), register='shell_c
25.             #dict( action=dict( module='debug', args=dict( msg='{{ shell_out}}' )))
26.     ]
27. )
28. #上面导入的对象，play_source执行的任务有哪些，变量到的分析
29. play = Play().load( play_source, variable_manager=variable_manager, loader=loader)

```

创建实例并执行命令

```

01. tqm = None
02. try:
03.     #tqm是taskQueueManager任务管理器生成的实例
04.     tqm = TaskQueueManager(
05.         inventory=inventory, #主机清单
06.         variable_manager=variable_manager, #参数管理
07.         loader=loader, #json等语法分析
08.         options=options, #选项
09.         passwords=passwords, #密码
10.     )
11.     result = tqm.run( play ) # tqm实例中的run方法开始执行play中的任务
12. finally:
13.     if tqm is not None: #如果tqm不为none
14.         tqm.cleanup() #清理
15.     shutil.rmtree( C.DEFAULT_LOCAL_TMP, True) #删除ansible执行任务是生成的临时目录

```

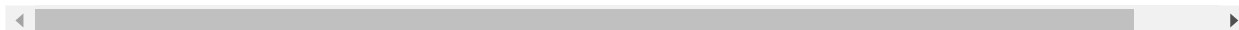
步骤二：测试执行脚本

```

01. [ root@localhost ~] # python ansible_adhoc.py
02.
03. PLAY [ Ansible Play ] *****
04. *****
05. TASK [ yum ] ***** Top *****
06. *****
07. ok: [ node2.tedu.cn]

```

08. ok: [node3.tedu.cn]
09. 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

01. [root@localhost ~] # vim run_pb.py
- 02.
03. #! /usr/bin/env python
04. # coding: utf8

导入模块

01. from collections import namedtuple
02. from ansible.parsing.data_loader import DataLoader
03. from ansible.vars.manager import VariableManager
04. from ansible.inventory.manager import Inventory Manager
05. from ansible.executor.playbook_executor import PlaybookExecutor

设置参数

01. # 初始化需要的对象
02. Options = namedtuple(
03. 'Options',
04. [
05. 'connection',
06. 'remote_user',
07. 'ask_sudo_pass',
08. 'verbosity',
09. 'ask_pass',

[Top](#)


```
10.     'module_path',
11.     'forks',
12.     'become',
13.     'become_method',
14.     'become_user',
15.     'check',
16.     'listhosts',
17.     'listtasks',
18.     'listtags',
19.     'syntax',
20.     'sudo_user',
21.     'sudo',
22.     'diff'
23. ]
24. )
25. # 初始化需要的对象
26. ops = Options(
27.     connection='smart',
28.     remote_user=None,
29.     ask_pass=None,
30.     sudo_user=None,
31.     forks=5,
32.     sudo=None,
33.     ask_sudo_pass=False,
34.     verbosity=5,
35.     module_path=None,
36.     become=None,
37.     become_method=None,
38.     become_user=None,
39.     check=False,
40.     diff=False,
41.     listhosts=None,
42.     listtags=None,
43.     listtasks=None,
44.     syntax=None
45. )
46. # 用来加载解析yaml文件或JSON内容,并且支持vault的解密
47. loader = DataLoader()
48. # 设置密码,需要是dict类型
49. passwords = dict()
50. # 根据inventory 加载对应变量的,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.         # play books就填写yml文件
03.         play_book = PlaybookExecutor(
04.             play_books=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 = play_book.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.     *****
07.     ok: [ node2.tedu.cn ]
08.     ok: [ node3.tedu.cn ]

```

[Top](#)

```

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

```

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 my lib] # vim my lib/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

```

01. def main():
02.     ##使用AnsibleModule类中的argument_spec来接收yuan、mudi两个参数，参数必须提供并
03.     mokuai = AnsibleModule(
04.         argument_spec=dict(
05.             yuan=dict(required=True, type='str'),
06.             mudi=dict(required=True, type='str')
07.         )
08.     )

```

执行动作

```

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

```

返回结果

```

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

```

编写主程序代码

```

01. if __name__ == '__main__':
02.     main()
03. [root@localhost myansi] # ansible dbbservers - m rcopy - a "yuan=/etc/hosts mudi=/opt"
04. node1.tedu.cn | SUCCESS => {
05.     "change" : true ,
06.     "changed" : false
07. }

```

[Top](#)