

NSD ARCHITECTURE DAY02

1. [练习1：playbook练习](#)
2. [案例2：变量练习](#)
3. [案例3：handlers练习](#)
4. [案例4：编写playbook](#)

1 练习1：playbook练习

1.1 问题

本案例要求：

- 安装Apache并修改监听端口为8080
- 修改ServerName配置，执行apachectl -t命令不报错
- 设置默认主页hello world
- 启动服务并设开机自启

1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：playbook的ping脚本检测

```

01. [ root@ansible ansible] # vim ping.yml
02. ---
03. - hosts: all
04.   remote_user: root
05.   tasks:
06.     - ping:
07. [ root@ansible ansible] # ansible-playbook ping.yml //输出结果
08.
09. PLAY [all] *****
10.
11. TASK [Gathering Facts] *****:
12. ok: [web1]
13. ok: [web2]
14. ok: [cache]
15. ok: [db1]
16. ok: [db2]
17.
18. TASK [ping] *****
19. ok: [db1]
20. ok: [web2]

```

[Top](#)

```

21.  ok: [ cache]
22.  ok: [ web1]
23.  ok: [ db2]
24.
25.  PLAY RECAP *****
26.  cache           : ok=2  changed=0  unreachable=0  failed=0
27.  db1             : ok=2  changed=0  unreachable=0  failed=0
28.  db2             : ok=2  changed=0  unreachable=0  failed=0
29.  web1            : ok=2  changed=0  unreachable=0  failed=0
30.  web2            : ok=2  changed=0  unreachable=0  failed=0

```

注意：如果检测的时候出错，会在当前的目录生成一个新的文件（以.retry结尾），可以去这个文件里面看是哪个主机的错

步骤二：用playbook安装Apache,修改端口，配置ServerName，修改主页，设置开机自启

```

01.  [ root@ansible ansible] # vim http.yml
02.
03.  ---
04.  - hosts: cache
05.    remote_user: root
06.    tasks:
07.      - name: install one specific version of Apache
08.        yum:
09.          name: httpd           //安装Apache
10.          state: installed
11.      - lineinfile:
12.          path: /etc/httpd/conf/httpd.conf
13.          regexp: '^Listen '
14.          line: 'Listen 8080'    //修改端口为8080
15.      - replace:
16.          path: /etc/httpd/conf/httpd.conf
17.          regexp: '^#(ServerName).*' //配置ServerName
18.          replace: '\1localhost'
19.      - service:
20.          name: httpd
21.          enabled: yes           //开机自启
22.          state: restarted
23.      - copy:
24.          src: /root/index.html  //修改主页，可以自己写个页面

```

[Top](#)

```

25.      dest: /var/www/html/index.html
26.
27.  [ root@ansible ansible] # curl 192.168.1.56:8080
28.  hello world
29.  [ root@ansible ansible] # ssh cache
30.  Last login: Fri Sep  7 09:32:05 2018 from 192.168.1.51
31.  [ root@cache ~] # apachectl -t
32.  Syntax OK

```

2 案例2：变量练习

2.1 问题

本案例要求熟悉playbook进阶：

- 练习使用user模块添加用户
- 练习使用变量简化task，让play通用性更强
- 练习使用过滤器

2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：使用user模块添加用户，并修改密码

```

01.  [ root@ansible ansible] # vim user.yml
02.  ---
03.  - hosts: cache
04.    remote_user: root
05.    vars:
06.      username: xiaoming
07.    tasks:
08.      - name: create user "{{ username }}"
09.        user: group=wheel uid=1000 name={{ username }}
10.      - shell: echo 123456 | passwd --stdin xiaoming
11.      - shell: chage -d 0 {{ username }}
12.  [ root@ansible ansible] # ansible-playbook user.yml //执行结果
13.
14.  PLAY [cache] *****
15.
16.  TASK [Gathering Facts] *****
17.  ok: [cache]
18.
19.  TASK [create user "xiaoming"] *****

```

[Top](#)

```

20.  changed: [ cache]
21.
22.  TASK [ command] *****
23.  changed: [ cache]
24.
25.  TASK [ command] *****
26.  changed: [ cache]
27.
28.  PLAY RECAP *****
29.  cache                : ok=4  changed=3  unreachable=0  failed=0

```

步骤二：变量过滤器，创建一个用户，设置密码

```

01.  [ root@ansible ansible] # vim user1.yml
02.  ---
03.  - hosts: cache
04.    remote_user: root
05.    tasks:
06.      - user:
07.        name: lisi
08.        group: root
09.        password: "{{ '123456' | password_hash( 'sha512') }}"
10.      - shell: chage - d 0 lisi
11.  [ root@ansible ansible] # ansible-playbook user1.yml
12.
13.  PLAY [ cache] *****
14.
15.  TASK [ Gathering Facts] *****
16.  ok: [ cache]
17.
18.  TASK [ user] *****
19.  changed: [ cache]
20.
21.  TASK [ command] *****
22.  changed: [ cache]
23.
24.  PLAY RECAP *****
25.  cache                : ok=3  changed=2  unreachable=0  failed=0

```

[Top](#)

步骤三：定义一个变量创建用户

```

01. [ root@ansible ansible] # vim user2.yml
02.
03. ---
04. - hosts: cache
05.   remote_user: root
06.   vars:
07.     user: zhangs
08.   tasks:
09.     - user:
10.       name: "{{user}}"
11.       group: root
12.       password: "{{'123456' | password_hash('sha512')}}"
13.     - shell: chage - d 0 "{{user}}"
14. [ root@ansible ansible] # ansible-playbook user2.yml
15. PLAY [cache] *****
16.
17. TASK [Gathering Facts] *****
18. ok: [cache]
19.
20. TASK [user] *****
21. changed: [cache]
22.
23. TASK [command] *****
24. changed: [cache]
25.
26. PLAY RECAP *****
27. cache                : ok=3  changed=2  unreachable=0  failed=0

```

3 案例3：handlers练习

3.1 问题

本案例要求：

- 安装Apache软件
- 配置文件，重新载入配置文件让服务生效
- 使用handlers来实现

[Top](#)

3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：error

playbook从上往下顺序执行，若报错，后面的命令不会在执行，若想解决有两种方法：

1) 当返回值为假时，显示true：- shell: setenforce 0 || true

```

01. [root@ansible ansible] # vim user5.yml
02. ---
03. - hosts: cache
04.   remote_user: root
05.   vars:
06.     user: bb
07.   tasks:
08.     - shell: setenforce 0 || true
09.     - user:
10.       name: "{{ user }}"
11.       group: root
12.       password: "{{ '123456' | password_hash('sha512') }}"
13.     - shell: chage -d 0 "{{ user }}"
14.
15. [root@ansible ansible] # ansible-playbook user5.yml
16.
17. PLAY [cache] *****
18.
19. TASK [Gathering Facts] *****
20. ok: [cache]
21.
22. TASK [command] *****
23. changed: [cache]
24.
25. TASK [user] *****
26. changed: [cache]
27.
28. TASK [command] *****
29. changed: [cache]
30.
31. PLAY RECAP *****
32. cache : ok=4 changed=3 unreachable=0 failed=0

```

[Top](#)

2、忽略：ignoring_errors: True(推荐使用这个，会有报错信息，告诉你错误忽略，继续执行下面的命令)

```

01. [ root@ansible ansible] # vim user6.yml
02. ---
03. - hosts: cache
04.   remote_user: root
05.   vars:
06.     user: bb
07.   tasks:
08.     - shell: setenforce 0
09.     ignore_errors: True
10.     - user:
11.       name: "{{ user }}"
12.       group: root
13.       password: "{{ '123456' | password_hash('sha512') }}"
14.     - shell: chage - d 0 "{{ user }}"
15.
16. [ root@ansible ansible] # ansible-playbook user6.yml
17.
18. PLAY [cache] *****
19.
20. TASK [Gathering Facts] *****
21. ok: [cache]
22.
23. TASK [command] *****
24. fatal: [cache]: FAILED! => {"changed": true, "cmd": "setenforce 0", "delta": "0:00:00.000000", "rc": 1, "results": ["setenforce: error: SELinux is disabled"], "start": "2019-01-04T08:58:03.123456Z", "stderr": "setenforce: error: SELinux is disabled", "stderr_lines": ["setenforce: error: SELinux is disabled"], "stdout": "", "stdout_lines": []}
25. ...ignoring
26.
27. TASK [user] *****
28. changed: [cache]
29.
30. TASK [command] *****
31. changed: [cache]
32.
33. PLAY RECAP *****
34. cache                : ok=4  changed=3  unreachable=0  failed=0

```

步骤二：handlers

关注的资源发生变化时采取的操作

[Top](#)

1) 使用handlers来配置文件，重新载入配置文件让服务生效

```

01. [root@ansible ansible] # vim adhttp.yml
02. ---
03. - hosts: cache
04.   remote_user: root
05.   tasks:
06.     - copy:
07.       src: /root/httpd.conf
08.       dest: /etc/httpd/conf/httpd.conf
09.       owner: root
10.       group: root
11.       mode: 0644
12.     notify:
13.       - restart httpd
14.   handlers:
15.     - name: restart httpd
16.       service: name=httpd state=restarted
17.
18. [root@ansible ansible] # ansible-playbook adhttp.yml
19.
20. PLAY [cache] *****
21.
22. TASK [Gathering Facts] *****
23. ok: [cache]
24.
25. TASK [copy] *****
26. ok: [cache]
27.
28. PLAY RECAP *****
29. cache                : ok=2  changed=0  unreachable=0  failed=0
30.
31. [root@ansible ansible] # ssh cache apachectl -t
32. Syntax OK
33. [root@ansible ansible] # curl 192.168.1.56:8080
34. hello world

```

2) 使用脚本调用变量更改服务

```

01. [root@ansible ansible] # vim adhttp2.yml
02. ---

```

[Top](#)


```

03.   - hosts: cache
04.     remote_user: root
05.     vars:
06.       server: httpd
07.     tasks:
08.       - copy:
09.         src: /root/httpd.conf
10.         dest: /etc/httpd/conf/httpd.conf
11.         owner: root
12.         group: root
13.         mode: 0644
14.       notify:
15.         - restart "{{ server }}"
16.     handlers:
17.       - name: restart "{{ server }}"
18.         service: name=httpd state=restarted
19. [ root@ansible ansible] # ansible-playbook adhttp2.yml
20.
21. PLAY [cache] *****
22.
23. TASK [Gathering Facts] *****
24. ok: [cache]
25.
26. TASK [copy] *****
27. ok: [cache]
28.
29. PLAY RECAP *****
30. cache                : ok=2  changed=0  unreachable=0  failed=0
31.
32. [ root@ansible ansible] #

```

4 案例4：编写playbook

4.1 问题

本案例要求：

- 把所有监听端口是8080的Apache服务全部停止

4.2 步骤

[Top](#)

实现此案例需要按照如下步骤进行。

步骤一：把监听端口是8080的Apache服务全部停止

```

01. [ root@ansible ansible] # vim ad.yml
02. ---
03. - hosts: cache
04.   remote_user: root
05.   tasks:
06.     - shell: netstat - atunlp | awk '{ print $4}' | awk '- F: ' '{ print $2}'
07.       register: result
08.     - service:
09.       name: httpd
10.       state: stopped
11. [ root@ansible ansible] # ansible-playbook ad.yml
12.
13. PLAY [ cache] *****
14.
15. TASK [ Gathering Facts] *****
16. ok: [ cache]
17.
18. TASK [ command] *****
19. changed: [ cache]
20.
21. TASK [ service] *****
22. changed: [ cache]
23.
24. PLAY RECAP *****
25. cache                : ok=3  changed=2  unreachable=0  failed=0

```

步骤二：when条件判断

1) 当系统负载超过0.7时，则关掉httpd

```

01. [ root@ansible ansible] # vim when.yml
02. ---
03. - hosts: cache
04.   remote_user: root
05.   tasks:
06.     - shell: uptime | awk '{ printf( "%2f", $( NF- 2) ) }'
07.       register: result
08.     - service:
09.       name: httpd

```

[Top](#)

```

10.         state: stopped
11.         when: result.stdout|float > 0.7
12.
13. [ root@ansible ansible] # ansible-playbook when.yml
14.
15. PLAY [cache] *****
16.
17. TASK [Gathering Facts] *****
18. ok: [cache]
19.
20. TASK [command] *****
21. changed: [cache]
22.
23. TASK [service] *****
24. changed: [cache]
25.
26. PLAY RECAP *****
27. cache                : ok=3  changed=2  unreachable=0  failed=0

```

步骤三：with_items标准循环

1) 为不同用户定义不同组

```

01. [ root@ansible ansible] # vim add.yml
02.
03. ---
04. - hosts: web2
05.   remote_user: root
06.   tasks:
07.     - user:
08.       name: "{{ item.name }}"
09.       group: "{{ item.group }}"
10.       password: "{{ '123456' | password_hash('sha512') }}"
11.       with_items:
12.         - { name: "aa", group: "users" }
13.         - { name: "bb", group: "mail" }
14.         - { name: "cc", group: "wheel" }
15.         - { name: "dd", group: "root" }
16. [ root@ansible ansible] # ansible-playbook add.yml
17.

```

[Top](#)

```

18.  PLAY [ web2 ] *****
19.
20.  TASK [ Gathering Facts ] *****:
21.  ok: [ web2 ]
22.
23.  TASK [ user ] *****
24.  changed: [ web2 ] => ( item={ u'group': u'users', u'name': u'aa' })
25.  changed: [ web2 ] => ( item={ u'group': u'mail', u'name': u'bb' })
26.  changed: [ web2 ] => ( item={ u'group': u'wheel', u'name': u'cc' })
27.  changed: [ web2 ] => ( item={ u'group': u'root', u'name': u'dd' })
28.
29.  PLAY RECAP *****
30.  web2                : ok=2  changed=1  unreachable=0  failed=0

```

2) 嵌套循环，循环添加多用户

```

01.  [ root@ansible ansible ] # vim add1.yml
02.  ---
03.  - hosts: web2
04.    remote_user: root
05.    vars:
06.      un: [ a, b, c ]
07.      id: [ 1, 2, 3 ]
08.    tasks:
09.      - name: add users
10.        shell: echo { { item } }
11.        with_nested:
12.          - "{ { un } } "
13.          - "{ { id } } "
14.
15.  [ root@ansible ansible ] # ansible-playbook add1.yml
16.
17.  PLAY [ web2 ] *****
18.
19.  TASK [ Gathering Facts ] *****:
20.  ok: [ web2 ]
21.
22.  TASK [ add users ] *****
23.  changed: [ web2 ] => ( item=[ u'a', 1 ] )

```

[Top](#)

```

24.  changed: [ web2 ] => ( item=[ u'a', 2 ] )
25.  changed: [ web2 ] => ( item=[ u'a', 3 ] )
26.  changed: [ web2 ] => ( item=[ u'b', 1 ] )
27.  changed: [ web2 ] => ( item=[ u'b', 2 ] )
28.  changed: [ web2 ] => ( item=[ u'b', 3 ] )
29.  changed: [ web2 ] => ( item=[ u'c', 1 ] )
30.  changed: [ web2 ] => ( item=[ u'c', 2 ] )
31.  changed: [ web2 ] => ( item=[ u'c', 3 ] )
32.
33.  PLAY RECAP *****
34.  web2                : ok=2  changed=1  unreachable=0  failed=0

```

步骤四：tags给指定的任务定义一个调用标识

1) tags 样例

```

01.  [ root@ansible ansible ] # vim adhttp.yml
02.  ---
03.  - hosts: cache
04.    remote_user: root
05.    tasks:
06.      - copy:
07.        src: /root/httpd.conf
08.        dest: /etc/httpd/conf/httpd.conf
09.        owner: root
10.        group: root
11.        mode: 0644
12.        tags: config_httpd
13.        notify:
14.          - restart httpd
15.      handlers:
16.        - name: restart httpd
17.          service: name=httpd state=restarted

```

2) 调用方式

```

01.  [ root@ansible ansible ] # ansible-playbook adhttp.yml --tags=config_httpd Top
02.
03.  PLAY [ cache ] *****

```

```

04.
05. TASK [ Gathering Facts ] *****
06. ok: [ cache ]
07.
08. TASK [ copy ] *****
09. ok: [ cache ]
10.
11. PLAY RECAP *****
12. cache                : ok=2  changed=0  unreachable=0  failed=0

```

3) include and roles

在编写playbook的时候随着项目越来越大，playbook越来越复杂。可以把一些play、task 或 handler放到其他文件中，通过包含进来是一个不错的选择

roles像是加强版的include，它可以引入一个项目的文件和目录

一般所需的目录层级有

vars : 变量层

tasks : 任务层

handlers : 触发条件

files : 文件

template : 模板

default : 默认，优先级最低

```

01. ...
02. tasks:
03.   - include: tasks/setup.yml
04.   - include: tasks/users.yml user=plj
05.   //users.yml 中可以通过{{ user }}来使用这些变量
06. handlers:
07.   - include: handlers/handlers.yml

```

步骤五：debug检测

```

01. [ root@ansible ansible ] # ansible-playbook --syntax-check http.yml //检测语法
02.
03. play book: http.yml
04. [ root@ansible ansible ] # ansible-playbook -C http.yml //测试运行
05.
06. [ root@ansible ansible ] # ansible-playbook http.yml --list-tasks

```

[Top](#)

```

07. //显示要执行的工作
08.
09. play book: http.yml
10.
11.   play #1( cache): cache   TAGS: []
12.     tasks:
13.       install one specific version of Apache   TAGS: []
14.       lineinfile   TAGS: []
15.       replace   TAGS: []
16.       service   TAGS: []
17.       copy   TAGS: []
18.
19.
20. [ root@ansible ansible] # vim debug.yml
21. ---
22. - hosts: cache
23.   remote_user: root
24.   tasks:
25.     - shell: uptime | awk '{ printf( "%f\n", $(NF-2)) }'
26.       register: result
27.     - shell: touch /tmp/isreboot
28.       when: result.stdout|float > 0.5
29.     - name: Show debug info
30.       debug: var=result
31.
32. [ root@ansible ansible] # ansible-playbook debug.yml //运行
33.
34. PLAY [ cache] *****
35.
36. TASK [ Gathering Facts] *****
37. ok: [ cache]
38.
39. TASK [ command] *****
40. changed: [ cache]
41.
42. TASK [ command] *****
43. skipping: [ cache]
44.
45. TASK [ Show debug info] *****
46. ok: [ cache] => {
47.   "result": {

```

[Top](#)

```
48.     "changed": true,
49.     "cmd": "uptime | awk '{ printf( \"%f \\n\", $( NF- 2 ) ) } '",
50.     "delta": "0: 00: 00.005905",
51.     "end": "2018- 09- 07 12: 57: 51.371013",
52.     "failed": false,
53.     "rc": 0,
54.     "start": "2018- 09- 07 12: 57: 51.365108",
55.     "stderr": "",
56.     "stderr_lines": [],
57.     "stdout": "0.000000",
58.     "stdout_lines": [
59.         "0.000000"
60.     ]
61. }
62. }
63.
64. PLAY RECAP *****
65.  cache           : ok=3  changed=1  unreachable=0  failed=0
```

[Top](#)