**15Architecture\_02playbook基础+playbook进阶**

**一 playbook基础-基础知识**

**1.1 ansible七种武器**

1.1.1 ansible命令,用于执行临时性的工作,必须掌握

1.1.2 ansible-doc是ansible模块的文档说明,针对每个模块都有详细的说明及应用案例介绍,功能和Linux系统man命令类似,必须掌握

1.1.3 ansible-console是ansible为用户提供的交互式工具,用户可以在ansible-console虚拟出来的终端上像shell一样使用ansible内置的各种命令,这为习惯使用shell交互方式的用户提供了良好的使用体验

1.1.4 ansible-galaxy从github上下载管理roles的一款工具,与Python的pip类似

1.1.5 ansible-playbook是日常应用中使用频率最高的命令,工作机制:通过读取先编好的playbook文件实现批量管理,可以理解为按一定条件组成的ansible任务集,必须掌握

1.1.6 ansible-vault主要用于配置文件加密,如编写的playbook文件中包含敏感信息.不想其他人随意查看,可用它加密\解密这个文件

1.1.7 ansible-pull

ansible-pull有两种工作模式pull/push,默认使用push模式工作,pull和push的工作模式机制相反

使用场景:有大批量机器需要配置,即便使用高并发线程依旧要花费很多时间

通常在配置大批量机器的场景下使用,灵活性稍有欠缺,但效率几乎可以无限提升,对运维人员的技术水平和前瞻性规划有较高要求

**1.2 JSON简介**

JSON是什么:

是Javascript对象表示法,它是一种基于文本 独立于语言的轻量级数据交换格式

JSON中的分隔符限于单引号,小括号,中括号,大括号,冒号,逗号

JSON特性:

JSON是纯文本

JSON具有”自我描述性”(人类可读)

JSON具有层级结构(值中存在值)

JSON可通过JavaScript进行解析

JSON语法规则

数据在名称/键值对中

数据由逗号分隔

大括号保存对象(键值对)

中括号保存数组

JSON数据的书写格式是: 名称/键值对

名称/键值对包括字段名称(在双引号中),后面写一个冒号,然后是值,例如:

“诗仙”:”李白”

JSON语法规则之数组

{“诗人”:

[”李白”,”杜甫”,”白居易”,”李贺”]

}

复合复杂类型

{“诗人”:

[

{“李白”:”诗仙”,”年代”:”唐”},

{“杜甫”:”诗圣”,”年代”:”唐”},

{“白居易”:”诗魔”,”年代”:”唐”},

{“李贺”:”诗鬼”,”年代”:”唐”}

]

}

**1.3 YAML简介**

YAML什么

是一个可读性高,用来表达数据序列的格式

YMAL ain’t Markup language

YAML基础语法

YMAL的结构通过空格来展示

数组使用”- ”来表示(-后跟了一个空格)

键值对使用”: ”来表示(:后跟了一个空格)

YAML使用一个固定的缩进风格表示数据层级结构关系

一般每个缩进级别由两个以上的空格组成

#表示注释

注意:不能使用tab键;同一层级缩进必须对齐

YAML的键值对表示方法:

采用冒号分隔; :后必须有一个空格

例子:

“诗仙”: “李白” #冒号后有个空格

或

“李白”: #冒号后有个空格

“诗仙” #缩进2个空格

复杂YAML的键值对嵌套

“诗人”: #冒号后有个空格

“李白”: “诗仙” #冒号后有个空格

或

“诗人”: #冒号后有个空格

“李白”: #冒号后有个空格

“诗仙” #冒号后有个空格

YAML数组表示方法:

使用一个-加一个空格

YAML数组例子:

- “李白”

- “杜甫”

- “白居易”

- “李贺”

哈希数组复合表达式:

“诗人”: #冒号后面有个空格

- “李白” #缩进两个空格,- 后面有个空格

- “杜甫”

- “白居易”

- “李贺”

复合复杂类型

“诗人”:

- #-号后面有个空格,下同

“李白”: ”诗仙” #冒号后面有个空格,下同

”年代”: ”唐”

-

“杜甫”: ”诗圣”

”年代”: ”唐”

-

“白居易”: ”诗魔”

”年代”: ”唐”

-

“李贺”: ”诗鬼”

”年代”: ”唐”

**1.4 Jinja2模板简介**

Jinja2是什么

Jinja2是基于Python的模板引擎,包含变量和表达式两部分,两者在模板求值时会被替换为值,模板中还有标签,控制模板的逻辑

为什么要学习Jinja2模板

playbook的模板使用Python的Jinja2模块来处理

Jinja2模板基本语法

模板的表达式都是包含在分隔符”{{ }}”内的

控制语句都是包含在分隔符”{% %}”内的

模板支持注释,都是包含在分隔符”{# #}”内,支持块注释

调用变量

{{varname}}

计算

{{2+3}}

判断

{{i in [1,2,3]}}

Jinja2模板控制语句

{% if name == ‘诗仙’ %}

李白

{% elif name == ‘诗圣’ %}

杜甫

{%elif name == ‘诗魔’ %}

白居易

{% else %}

李贺

{% endif %}

Jinja2模板控制语句

{% if name == ...... %}

......

{% elif name == ‘于谦’ %}

{% for method in [抽烟,喝酒,烫头] %}

{{do method}}

{% endfor %}

... ...

{% endif %}

Jinja2过滤器

变量可以通过过滤器修改.过滤器与变量用管道符号 | 分隔,也可以用圆括号

传递可选参数,多个过滤器可以链式调用,前一个过滤器的输出会被作为后一个 过滤器的输入

例如:

加密一个字符串: {{‘astr’ | password\_hash(‘sha512’)}}

过滤器在线文档:http://docs.jinkan.org/docs/jinja2/templates.html#builtin-filters

**二 playbook基础-playbook**

**2.1 playbook是什么**

playbook是什么

playbook是ansible用于配置,部署和管理托管主机的剧本,通过playbook的详细描述,执行其中的一系列tasks,可以让远端主机达到预期状态

也可以说,playbook字面意思即剧本,现实中由演员按剧本表演,在ansible中由计算机进行安装,部署应用,提供对外服务,以及组织计算机处理各种各样的事情

为什么要使用playbook

执行一些简单的任务,使用ad-hoc命令可以方便的解决问题,但有时一个设施过于复杂时,执行ad-hoc命令是不合适的,最好使用playbook

playbook可以反复使用编写好的代码,可以放到不同的机器上面,像函数一样,最大化的利用代码,在使用ansible的过程中,处理大部分操作都是在编写playbook

**2.2 playbook语法基础**

playbook由YAML语言编写,遵循YAML标准

在同一行中,#之后的内容表示注释

用一个列表中的元素应该保持相同的缩进,不能使用TAB键

playbook由一个或多个play组成

play中hosts,variables,roles,tasks等对象的表示方法都是键值中间:分隔表示

YAML文件开始行为---,是固定格式

playbook构成

hosts: 定义将要执行playbook的远程主机组

vars: 定义playbook运行时需要使用的变量

tasks: 定义将要在远程主机上执行的任务列表

handlers: 定义task执行完成后需要调用的任务

name: tasks下一级使用,对tasks的描述

playbook执行结果

使用ansible-playbook运行playbook文件,输出内容为JSON格式,由不同颜色组成便于识别: 绿色表示执行成功,黄色代表系统状态发生改变,红色表示执行失败

第一个playbook

ansible40 ~]# vim ping.yml

--- #第一行

- hosts: all

remote\_user: root #远程用户账户名

tasks:

- ping:

ansible-playbook ping.yml -f 5

-f 并发进程数量,默认是5,最多不超过CPU核数的2倍

hosts行,内容是一个(多个)组或主机的patterns,用逗号分隔

tasks:

命令的集合;

每一个play包含了一个task列表(任务列表);

一个task在其所对应的所有主机上(通过host pattern匹配的主机)执行 完毕后,下一个task才会执行

hosts: 主机的集合,定义要执行任务的主机

第二个playbook: 给web主机添加用户zhangsan ,密码123,添加到users组

编写playbook脚本:

ansible40 ~]# vim user.yml

---

- hosts: web

remote\_user: root

tasks:

- name: adduser user zhangsan

user:

name: zhangsan

group: users

- name: change zhangsan password

shell: echo 123 | passwd --stdin zhangsan

执行:

ansible40 ~]# ansible-playbook user.yml

验证:

web141 ~]# id zhangsan

uid=1001(zhangsan) gid=100(users) 组=100(users)

案例:使用playbook编写脚本,实现所有web主机安装apache;修改配置文件的监听端口为8080;设为默认首页为hello world;启动服务;设置开机自启

编写playbook脚本

ansible40 ~]# vim apache.ymal

---

- hosts: web

remote\_user: root

tasks:

- name: install apache on web

yum:

name: httpd

state: installed

- lineinfile:

path: /etc/httpd/conf/httpd.conf

regexp: '^Listen '

line: 'Listen 8080'

- service:

name: httpd

enabled: yes

state: restarted

- name: set httpd default page

copy:

src: /root/index.html

dest: /var/www/html/index.html

创建index.html首页

ansible40 ~]# vim index.html

hello world

执行ansible-playbook命令

ansible40 ~]# ansible-playbook apache.ymal

验证

ansible40 ~]# curl http://192.168.1.42:8080

hello world!

**三 playbook进阶-语法进阶**

**3.1 变量**

案例: 使用变量,编写playbook脚本,在db主机中添加l4用户,密码123

ansible40 ~]# vim user.yml

---

- hosts: db

remote\_user: root

vars:

username: l4

tasks:

- name: adduser user "{{username}}"

user:

name: "{{username}}"

group: users

- name: change "{{username}}" password

shell: echo 123 | passwd --stdin "{{username}}"

ansible40 ~]# ansible-playbook user.yml

db143 ~]# id l4

uid=1000(l4) gid=100(users) 组=100(users)

**3.1.1 设密码,解决密码明文问题**

user模块的password不能设置密码,因为password是把字符串直接写入shadow文件,而Linux的shadow密码是经过加密的.

解决方案:

变量过滤器password\_hash

{{‘密码字符串’ | password\_hash(‘sha512’)}}

案例: 给db主机添加用户plj,设置默认密码123qqq...A

ansible40 ~]# vim userplj.yml

---

- hosts: db

remote\_user: root

vars:

username: plj

tasks:

- name: create user "{{username}}"

user:

password={{'123qqq...A' | password\_hash('sha512')}}

name={{username}}

ansible40 ~]# ansible-playbook userplj.yml

ansible40 ~]# ssh -X plj@192.168.1.43

plj@192.168.1.43's password: #密码输入123qqq...A

**3.1.2 变量传递参数**

-e 参数

参数格式必须是json或yaml

直接传参: ansible-playbook user.yml -e ‘{“username”:”wk”}’

yaml格式 可以使用参数文件,进行文件传参

ansible40 ~]# cp userplj.yml user1.yml

ansible40 ~]# vim user1.yml

此文件中没有vars语句定义变量,但后续语句有调用变量量,后续有语句调用变量

---

- hosts: db

remote\_user: root

tasks:

- name: create user "{{username}}"

user:

password={{'123qqq...A' | password\_hash('sha512')}}

name={{username}}

ansible40 ~]# vim args.yml #编辑yaml格式的传参文件

---

username:

nb

ansible40 ~]# ansible-playbook user1.yml -e @args.yml #传参

ansible40 ~]# ssh -X nb@192.168.1.43

nb@192.168.1.43's password: #密码输入123qqq...A

**3.2 error**

ansible-playbook对错误的处理

默认情况判断$?,如果值不为0就停止执行

但某些情况我们需要忽略错误继续执行

例如: 创建缓存目录,然后重启apache

ansible40 ~]# cp apache.yml http\_buff.yml

ansible40 ~]# vim http\_buff.yml

---

- hosts: web

remote\_user: root

tasks:

- name: mkdir for http buff dir

shell: mkdir /tmp/cache

**ignore\_errors: Ture**

- copy:

src: httpd.conf

dest: /etc/httpd/conf/httpd.conf

owner: root

group: root

mode: 0644

- service:

name: httpd

state: restarted

web141 ~]# scp /etc/httpd/conf/httpd.conf [root@192.168.1.40:/root](mailto:root@192.168.1.40:/root)

ansible40 ~]# ansible-playbook http\_buff.yml

错误处理方法:

ignore\_errors: 对错误的处理方式

True表示忽略错误继续执行

False表示遇到错误就停止执行,默认为False

**3.3 tags**

tags: 给指定的任务定义一个调用标识

使用格式:

name: NAME

module: arguments

tags: TAG\_ID

playbook调用方式

-t TAGS, --tags=TAGS

只运行配置文件中打了tags标签的模块

**3.4 handlers**

当关注的资源发生变化时采取的操作

nofity这个action可用于在每个play的最后被触发,这样可以避免有多次改变发生时每次都执行指定的操作,取而代之仅在所有变化发生完成后一次性的执行指定操作

在notify中列出的操作称为handler,即notify调用handler中定义的操作

与tasks平级

注意事项:

notify调用的是handler段的name定义的串,必须一致,否则达不到触发的效果

多个task触发同一个notify的时候,同一个服务只会触发一次

notify可以触发多个条件,在生产环境中往往涉及到某一个配置文件的改变需要重启若干各服务的场景,handler用到这里非常适合

可偶尔接vars可以写出非常普适的服务管理脚本

案例: 修改httpd的配置文件,端口改为80,重新载入配置文件让服务生效

ansible40 ~]# cp http\_buff.yml http\_buff1.yml

ansible40 ~]# vim http\_buff1.yml

---

- hosts: web

remote\_user: root

tasks:

- name: create cache dir

shell: mkdir /tmp/cache

ignore\_errors: True

- copy:

src: httpd.conf

dest: /etc/httpd/conf/httpd.conf

owner: root

group: root

mode: 0644

tags: webconf

notify: #调用

- restart web #与handlers下name定义的操作名称相同

handlers: #与tasks平级

- name: restart web #定义操作名称,必须定义

service:

name: httpd

state: restarted

ansible40 ~]# vim httpd.conf

Listen 80 #修改端口为80

ansible40 ~]# ansible-playbook http\_buff1.yml

验证:

ansible40 ~]# curl <http://192.168.1.41> #不添加端口号8080,默认为80

hello world!

**3.5 when**

有些时候需要在满足特定的条件后再触发某一项操作,或在特定的条件下终止某个行为,这个时候需要进行条件判断,when正式解决这个问题的最佳选择,远程中的系统变量facts作为when的条件,可以通过setup模块查看

ansible40 ~]# ansible cache45 -m setup

ansible40 ~]# ansible cache45 -m setup -a 'filter=ansible\_os\_family'

#查看系统类型的facts

---

- name: install vim

hosts: all

tasks:

- name: install vim via yum

yum: name=vim-enhanced state=installed

when: ansible\_os\_family == “Redhat”

- name: install vim via apt

apt: name=vim state=installed

when: ansible\_os\_family == “Debian”

**3.6 register**

有时候我们需要判断前一个命令的执行结果去处理后面的操作,这时候就需要register模块来保存前一个命令的返回状态,在后面进行调用

command: test command

register: result #将test command的返回状态保存到result中

command: run command

when: result #调用result中保存的返回状态

案例: 编写playbook,把系统负载太高的httpd服务停止

ansible40 ~]# vim load.yml

---

- hosts: web

tasks:

- name: show uptime info

shell: uptime | awk '{printf ("%.2f",$(NF-2))}'

register: result

- name: shutdown web server

service:

name: httpd

state: stopped

when: result.stdout | float > 0.7

**3.7 with\_items**

with\_items是playbook标准循环,可以用于迭代一个列表或字典,通过{{item}}获取每次迭代的值

案例:创建多个用户

---

- hosts: web

remote\_user: root

tasks:

- name: add users

user: group=wheel password={{‘123’ | password\_hash(‘sha512’)}} name={{item}}

with\_items: [“nb”,”dd”,”plj”,”lx”]

案例:为不同的用户定义不同组和密码

ansible40 ~]# vim useradd.yml

---

- hosts: db

tasks:

- name: adduser user "{{item.uname}}"

user:

name: "{{item.uname}}"

group: "{{item.grp}}"

password: "{{item.pwd | password\_hash('sha512')}}"

with\_items:

-

uname: nb

grp: users

pwd: "123"

-

uname: wk

grp: adm

pwd: banana

-

uname: dd

grp: daemon

pwd: book

-

uname: xx

grp: root

pwd: a

ansible40 ~]# ansible-playbook useradd.yml

db143 ~]# id nb\wk\dd\xx #验证

**3.8 with\_nested**

**3.9 include and roles**

在编写playbook的时候随着项目越来越大,playbook越来越复杂,修改也很麻烦.这时可以把一些play,task或handler放到其他文件中,通过include指令调用

tasks:

- include: tasks/setup.yml

- include: tasks/users.yml user=plj

#users.yml中可以通过{{user}}来使用这些变量

handlers:

- include: handlers/handlers.yml

roles像加强版的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

**四 playbook进阶-调试**

4.1 debug

检查语法: ansible-playbook --syntax-check playbook.yaml

测试运行: ansible-playbook -C playbook.ymal

显示受到影响的主机 --list-hosts

显示工作的task --list-tasks

显示将要运行的tag --list-tags

debug模块可以在运行时输出更为详细的信息,帮助我们排错

---

- hosts: web

tasks:

- name: show uptime info

shell: uptime | awk '{printf ("%.2f",$(NF-2))}'

register: result

- name: shutdown web server

service:

name: httpd

state: stopped

when: result.stdout | float > 0.7

- name: show debug info

debug: var=result