# ansible 01

**1、准备6台虚拟机**

1 ansible 管理机器

2 web1 托管机器

3 web2 托管机器

4 db1 托管机器

5 db2 托管机器

6 cache 托管机器

**2、配置免密登陆**

在物理主机上配置名称解析

[root@room8pc16 nsd2018]# for i in {1..6}

> do

> echo -e "192.168.4.$i\tnode$i.tedu.cn\tnode$i" >> /etc/hosts

> done

提前将服务器的密钥保存，不需要ssh时回答yes

[root@room8pc16 nsd2018]# ssh-keyscan node{1..6} >> /root/.ssh/known\_hosts

[root@room8pc16 nsd2018]#ssh-keygen -f /root/.ssh/id\_rsa -N ' ' #自动生成密钥

实现免密登陆

[root@room8pc16 nsd2018]# for i in {1..6}

> do

> ssh-copy-id node$i

> done

1. **配置一个 yum 源**

使用 CentOS-7-x86\_64-Everything-1511.iso

mount -t iso9660 -o ro,loop /ISO/CentOS-7-x86\_64-Everything-1511.iso /var/ftp/rhel7

或者

[root@room9pc01 public]# scp ansible\_soft.tar.xz 192.168.1.10:/root

#用yum挨个把包装上

在创建一个目录用于存放 ansible 的安装包文件

mkdir /var/ftp/ansible

拷贝安装文件到该目录，后创建索引

cd /var/ftp/ansible

createrepo .

配置 ansible 管理机器，添加 repo 文件

[ansible]

name=ansible

baseurl=ftp://192.168.4.254/ansible

enabled=1

gpgcheck=0

在 ansible 管理机器上

yum clean all

yum install -y ansible

安装完成以后执行，没有报错，正确显示版本即可

ansible --version

# 主机定义与分组

ansible 的配置文件是 ansible.cfg

ansible.cfg 的查找顺序是

• Ansible配置文件查找顺序

– 首先检测 ANSIBLE\_CONFIG 变量定义的配置文件

– 其次检查当前目彔下的 ./ansible.cfg 文件

– 再次检查当前用户家目彔下 ~/ansible.cfg 文件

– 最后检查 /etc/ansible/ansible.cfg 文件

• /etc/ansible/ansible.cfg 默认配置文件路径

• ansible.cfg 配置文件中 inventony 指定主机分组文件的路径和地址，默认分组文件 /etc/ansible/hosts

• 格式

– # 表示注释

[组名称]

主机名称或ip地址,登录用户名,密码、端口等信息

• 测试

– ansible [组名称] --list-hosts

## 主机定义与分组

• inventory 参数说明

– ansible\_ssh\_host

– 将要连接的进程主机名.不你想要设定的主机的别名不同的话,可通过此变量设置.

– ansible\_ssh\_port

– ssh端口号.如果丌是默认的端口号,通过此变量设置.

– ansible\_ssh\_user

– 默认的 ssh 用户名

• inventory 参数说明

– ansible\_ssh\_pass

– ssh 密码(这种方式并丌安全,我们强烈建议使用 --ask-pass 或 SSH 密钥)

– ansible\_sudo\_pass

– sudo 密码(建议使用 --ask-sudo-pass)

– ansible\_sudo\_exe (new in version 1.8)

– sudo 命令路径(适用亍1.8及以上版本)

• inventory 参数说明

– ansible\_connection

– 不主机的连接类型.比如:local, ssh 戒者 paramiko.Ansible 1.2 以前默认使用 paramiko.1.2 以后默认使用 'smart','smart' 方式会根据是否支持ControlPersist, 来判断'ssh' 方式是否可行.

– ansible\_ssh\_private\_key\_file

– ssh 使用的私钥文件.适用亍有多个密钥,而你不想使用SSH 代理的情况.

• inventory 参数说明

– ansible\_shell\_type

– 目标系统的shell类型.默认情况下,命令的执行使用 'sh'语法,可设置为 'csh' 戒 'fish'.

– ansible\_python\_interpreter

– 目标主机的 python 路径.适用亍的情况: 系统中有多个

Python, 戒者命令路径丌是"/usr/bin/python”

[root@ansible ~]# cd /etc/ansible/

[root@ansible ansible]# vim hosts

hosts 的配置

[web]

web[1:2]

[db]

db1

db2

[app:children] # 指定子组

web

db

[app:vars]

ansible\_ssh\_user="root"

ansible\_ssh\_pass="123456"

[other]

cache ansible\_ssh\_user="root" ansible\_ssh\_pass="123456"

[root@ansible ansible]# ansible web --list-hosts #测试

hosts (2):

web1

web2

# ansible 命令基础

ansible 主机分组 -m 模块 -a '命令和参数'

[root@ansible ansible]# ansible web -m ping

#检测是否能连接到web(绿色代表成功，红色可能是没有导入公钥)

[root@cache ~]# rm -rf /root/.ssh/authorized\_keys #实验，删除公钥

[root@ansible ansible]# vim hosts

……

[other]

cache ansible\_ssh\_user="root" ansible\_ssh\_pass="1"

[root@ansible ansible]# ansible other -m ping #变成绿色成功

[root@web1 ~]# rm -rf /root/.ssh/authorized\_keys

[root@web2 ~]# rm -rf /root/.ssh/authorized\_keys

[root@ansible ansible]# ansible web -m ping #报红

[root@ansible ansible]# vim /etc/ansible/hosts

……

[web:vars] **#vars为变量 #为这个组所有主机配置用户名和密码**

ansible\_ssh\_user="root"

ansible\_ssh\_pass="1"

[root@ansible ansible]#rm -rf /root/.ansible/cp/\* #如果有缓存的话，需要清除缓存

[root@ansible ansible]# vim hosts

……

[app:children]  **#子组定义**

web

db

[root@ansible ansible]# ansible app --list-hosts

## 自定义主机（应用）分组（一个组里同时有数据库和web）

[root@ansible ~]# cd /var/ #随便进一个目录

[root@ansible var]# mkdir ooxx

[root@ansible var]# cd ooxx/

[root@ansible ooxx]# touch ansible.cfg #声明 inventory在这里

[root@ansible ooxx]# vim /etc/ansible/ansible.cfg

10 [defaults]

14 #inventory = /etc/ansible/hosts **#复制这两行**

[root@ansible ooxx]# vim ansible.cfg

[defaults]

inventory = myhosts #当前目录下的文件

[root@ansible ooxx]# vim myhosts #创建自定义分组文件

[app1] #项目一

web1

db1

[app2] #项目二

web2

db2

ceche

[app:children] #总项目

app1

app2

[root@ansible ooxx]# ansible app --list-hosts #查看

[root@ansible ~]# cd /root

[root@ansible ~]# ansible app1 --list-hosts #报错

# 动态主机

[root@ansible ansible]# cd /var/ooxx/

[root@ansible ooxx]# touch aaa

[root@ansible ooxx]# chmod 755 aaa

[root@ansible ooxx]# vim aaa #编写脚本

#!/bin/bash

echo '

{

"web" : ["web1", "web2"],

"bb" : ["db1", "db2"],

"other" : ["cache"]

}'

[root@ansible ooxx]# vim ansible.cfg #修改文件

[defaults]

inventory = aaa

[root@ansible ooxx]# ansible web --list-hosts #查看

# 批量执行

ansible命令基础

• ansible <host-pattern> [options]

– host-pattern 主机戒定义的分组

– -M 指定模块路径

– -m 使用模块,默认 command 模块

– -a or --args 模块参数

– -i inventory 文件路径,戒可执行脚本

– -k 使用交亏式登彔密码

– -e 定义变量

– -v 详绅信息,-vvvv 开吭 debug 模式

**ansible命令基础**

• 列出要执行的主机,丌执行任何操作

– ansible all --list-hosts

• 批量检测主机

– ansible all -m ping

• 批量执行命令

– ansible all -m command -a 'id' -k

[root@ansible ooxx]# cd /etc/ansible/

[root@ansible ansible]# vim hosts

……

[other]

192.168.1.33 #删掉密码

[root@ansible ansible]# uptime #查看系统负载

16:25:17 up 6:10, 2 users, load average: 0.00, 0.01, 0.05

[root@ansible ansible]# ansible db -m command -a 'uptime' #-m批量执行命令

[root@ansible ansible]# ansible other -m command -a 'uptime' -k

#-k表示交互（问你密码）

**批量部署证书文件**

• 每次交亏输入密码比较麻烦

• 密码写入配置文件安全性很差

• 丌同主机丌同密码,配置文件要上天

• 使用 key 方式认证,是一个丌错的选择

• 给所有主机部署公钥

– ansible all -m authorized\_key -a "user=root

exclusive=true manage\_dir=true key='$(<

/root/.ssh/authorized\_keys)'" -k -v

[root@ansible ansible]# cd /root/.ssh/

[root@ansible .ssh]# cat id\_rsa.pub

创建密钥对 id\_rsa 是私钥， id\_rsa.pub 是公钥

ssh-keygen -t rsa -b 2048 -N ''

给所有主机部署密钥

[root@ansible .ssh]#ansible all -m authorized\_key -a "user=root exclusive=true manage\_dir=true key='$(< /root/.ssh/id\_rsa.pub)'" -k

[root@ansible ansible]# rm -rf /root/.ssh/known\_hosts #模拟报错

[root@ansible ansible]# ansible all -m authorized\_key -a "user=root exclusive=true manage\_dir=true key='$(< /root/.ssh/id\_rsa.pub)'" -k #报错

[root@ansible ansible]# vim /etc/ansible/ansible.cfg

61 host\_key\_checking = False #主机的key检测需要输入yes（去掉#）

[root@ansible ansible]# ansible all -m authorized\_key -a "user=root exclusive=true manage\_dir=true key='$(< /root/.ssh/id\_rsa.pub)'" -k

#成功，不成功的话，注意当前路径

# 模块

ansible-doc 查看帮助，必须掌握

ansible-doc -l 列出所有模块

ansible-doc 模块名 查看该模块的帮助信息

ping模块

没有参数，检测主机的连通性，与 ping 无关，主要检测 ssh 是否可以连接

## command | shell | raw 重点模块

command是默认模块，没有启用shell，所有shell相关特性命令无法使用，例如< > | &

raw 模块，没有 chdir create remove 等参数，能执行大部分操作

shell 模块，启动 /bin/sh 运行命令，可以执行所有操作

测试

ansible cache -m command -a 'chdir=/tmp touch f1' 创建成功

ansible cache -m shell -a 'chdir=/tmp touch f2' 创建成功

ansible cache -m raw -a 'chdir=/tmp touch f3' 文件可以创建，但无法切换目录，文件在用户家目录下生成

复杂操作怎么办，使用脚本来解决

#!/bin/bash

adduser zhang3 # 创建用户 zhang3

echo 123456 |passwd --stdin zhang3 # 修改密码

chage -d 0 zhang3 # 第一次登录必须修改密码

给app1分组添加用户li4

系统里没有zhang3用户就添加，如果zhang3存在就不添加

修改li4的默认密码123456

**#!/bin/bash**

**id zhang3**

**if [ $? != 0 ];then**

**useradd li4**

**echo 123456 |passwd --stdin li4**

**fi**

[root@ansible ansible]# ansible all -m script -a '1.sh' #执行脚本

## copy lineinfile replace 模块

copy 把文件发布到远程其他主机上面

lineinfile 修改一个文件的一行，以行为基础，整行修改

replace 修改文件的某一部分，以正则表达式匹配为基础修改

**利用 copy 模块** 修改所有机器的 /etc/resolv.conf 为

nameserver 8.8.8.8

[root@ansible ~]# ansible all -m copy -a 'src=/etc/resolv.conf dest=/etc/resolv.conf'

**利用 lineinfile模块** 修改 /etc/sysconfig/network-scriopts/ifcfg-eth0

ONBOOT=yes|no

ansible cache -m lineinfile -a 'path=/etc/sysconfig/network-scripts/ifcfg-eth0 regexp="^ONBOOT" line="ONBOOT=\"no\""

**利用 replace模块** 修改 /etc/sysconfig/network-scriopts/ifcfg-eth0

ONBOOT=no|yes

ansible cache -m replace -a 'path=/etc/sysconfig/network-scripts/ifcfg-eth0 regexp="^(ONBOOT=).\*" replace="\1\"yes\""' #\1表示第一个括号表示的内容

**yum 模块** installed 安装， removed 删除

ansible other -m yum -a 'name="lrzsz" state=removed'

ansible other -m yum -a 'name="lftp" state=removed'

ansible other -m yum -a 'name="lrzsz,lftp" state=installed'

**service 模块** name 指定服务名称，enabled= yes|no 设置开机启动， state=stopped|started 启动关闭服务

设置 chronyd 服务开启启动，并启动服务

ansible other -m service -a 'name="chronyd" enabled="yes" state="started"'

**setup 模块** 查看信息 filter 过滤指定的关键字

[root@ansible ansible]# ansible cache -m setup #查看系统信息、硬件信息

[root@ansible ansible]# ansible cache -m setup -a 'filter="ansible\_os\_family"' #查看关键字（系统）信息

[root@ansible ansible]# ansible all -m setup -a 'filter="ansible\_os\_family"'

#查看所有系统信息

**练习题**

**– 安装 apache**

**– 修改 apache 监听的端口为 8080**

**– 为 apache 增加 NameServer 配置**

**– 设置默认主页**

**– 启动服务**

**– 设置开机自启动**

安装一个apache

[root@ansible ~]# ansible cache -m yum -a 'name=httpd state=installed'

#装包

[root@ansible ~]# scp cache:/etc/httpd/conf/httpd.conf ./ #拷到当前目录

[root@ansible ~]# vim httpd.conf

42 Listen 8080 #将端口号改为8080

96 ServerName localhost #改为localhost

[root@ansible ~]# ansible cache -m copy -a 'src=httpd.conf dest=/etc/httpd/conf/httpd.conf' #拷贝到cache机器

[root@ansible ~]# ansible cache -m service -a 'name="httpd" enabled="yes" state="started"' #起服务

[root@cache ~]# apachectl -t #登陆到cache的机器上，查看

Syntax OK

[root@cache ~]# systemctl is-enabled httpd

enabled

[root@ansible ~]# ansible cache -m copy -a 'src=/var/www/html/index.html dest=/var/www/html/index.html' #设置默认主页

# playbook语法基础

[root@ansible ~]# cat ping.yml

---

- hosts: all

remote\_user: root

tasks:

- ping:

[root@ansible ~]# ansible-playbook ping.yml #启动脚本

编写playbook 实现以下效果

**– 安装 apache**

**– 修改 apache 监听的端口为 8080**

**– 为 apache 增加 NameServer 配置**

**– 设置默认主页**

**– 启动服务**

**– 设置开机自启动**

[root@ansible ~]# cat ping.yml

---

- hosts: db

remote\_user: root

tasks:

- user:

name: li4

groups: users

- name: ooxx setpasssword for user li4

shell: echo '123456' |passwd --stdin li4

- shell: chage -d 0 li4

**playbook httpd.yml**

---

- hosts: web

remote\_user: root

tasks:

- name: install the latest version of Apache

yum:

name: httpd

state: installed

- lineinfile:

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

regexp: '^Listen '

insertafter: '^#Listen '

line: 'Listen 8080'

- lineinfile:

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

regexp: '^#ServerName'

line: 'ServerName localhost'

- copy:

src: /root/index.html

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

owner: apache

group: apache

mode: 0644

- service:

name: httpd

state: started

enabled: yes

**使用变量和过滤器添加用户**

---

- hosts: cache

remote\_user: root

vars:

un: nb

tasks:

- user:

name: "{{un}}"

group: users

password: "{{'123456'|password\_hash('sha512')}}"

- name: ooxx

shell: chage -d 0 "{{un}}"

**添加用户，忽略错误**(**erro**)

---

- hosts: db

remote\_user: root

vars:

un: zhang3

tasks:

- shell: adduser "{{un}}"

ignore\_errors: True

- shell: echo 123456|passwd --stdin "{{un}}"

- name: ooxx

shell: chage -d 0 "{{un}}"

**handlers 触发** #与tasks对齐

---

- hosts: web

remote\_user: root

tasks:

- name: install the latest version of Apache

yum:

name: httpd

state: installed

- lineinfile:

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

regexp: '^Listen '

insertafter: '^#Listen '

line: 'Listen 8080'

notify:

- restart\_httpd

- lineinfile:

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

regexp: '^#ServerName'

line: 'ServerName localhost'

notify:

- restart\_httpd

- copy:

src: /root/index.html

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

owner: apache

group: apache

mode: 0644

- copy:

src: /root/httpd.conf

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

owner: root

group: root

mode: 0644

tags: config\_httpd

notify:

- restart\_httpd

handlers:

- name: restart\_httpd

service:

name: httpd

state: restarted

enabled: yes

**把负载高的web服务器停止 (when判断)**

---

- hosts: web

remote\_user: root

tasks:

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

register: result

- service: name=httpd state=stopped

when: result.stdout|float > 0.7

追加 debug 调试信息

- name: Show debug info

debug: var=ooxx

测试命令 awk 'BEGIN{while(1){}}' #死循环

**循环添加多用户**

---

- hosts: web2

remote\_user: root

tasks:

- user:

name: "{{item.name}}"

group: "{{item.group}}"

password: "{{'123456'|password\_hash('sha512')}}"

with\_items:

- {name: "nb", group: "users"}

- {name: "dd", group: "mail" }

- {name: "jj", group: "wheel"}

- {name: "lx", group: "root" }

**嵌套循环**

---

- hosts: cache

remote\_user: root

vars:

un: [a, b, c]

id: [1, 2, 3]

tasks:

- name: add users

shell: echo {{item}}

with\_nested:

- "{{un}}"

- "{{id}}"

标签设置 tags: config\_httpd

检测语法

ansible-playbook --syntax-check playbook.yaml

测试运行

ansible-playbook -C playbook.yaml

显示受到影响到主机 --list-hosts

显示工作的 task --list-tasks

显示将要运行的 tag --list-tags