部署环境

六台: ansible web11 web22 db1 db2 cache

192.168.1.40-45

真机

[student@room9pc01 ~]$ base-vm ansible web11 web22 db1 db2 cache

cd /linux-soft/04/

mkdir /var/ftp/public/

cp -r ansible/ /var/ftp/public/

createrepo --update .

**ansible基础**

**介绍**:

ansiable是2013年推出的一款IT自动化和DevOps软件,2015年被RedHat收购.是基于Python研发,糅合很多老运维工具的优点,实现了批量操作系统配置,批量程序部署,批量运行命令等功能

ansible可以实现:

--自动化部署APP

--自动化管理配置项

--自动化持续交付

--自动化(AWS)云服务管理

为什么选用ansible

--活跃度(社区)

--学习成本

--使用成本

--编码语言

--性能

--使用是否广发

优点:

--只需要SSH和Python即可使用

--无客户端

--ansible功能强大,模块丰富

--上手容易,门槛低

--基本Python开发,做二次开发更容易

--使用公司比较多,社区活跃

特性:

模块化设计,调用特定的模块完成特定任务

基于Python语言实现

--paramiko

--PyYAML

--Jinja2

其模块支持JSON等标准输出格式,可以采用任何编程语言重写

部署简单

主从模式工作

支持自定义模块

易于使用

支持多层部署

支持异构IT环境

软件依赖关系

1 对管理主机:

要求Python 2.6 或 Python2.7

ansible使用以下模块,都需要安装

--paramiko

--PyYAML

--Jinja2

--httplib2

--six

2 对于被托管主机

--ansible默认通过SSH协议管理机器

--被管理主机要开启ssh服务,允许ansible主机登录

--在托管节点上也需要安装Python2.5或以上的版本

如果托管节点上开启了SElinux,需要安装libselinux-python

Json (JavaScript Object Notation, JavaScript对象表示法),一种基于文本独立与语言的轻量级数据交换格式

ansible:

六台都做----------->v

v

[root@ansible ~]#vim /etc/yum.repos.d/local.repo

[local\_repo]

name=CentOS-$releasever - Base

baseurl="ftp://192.168.1.254/centos-1804"

enabled=1

gpgcheck=1

[ansible]

name=ansible

baseurl="ftp://192.168.1.254/public/ansible"

enabled=1

gpgcheck=0

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

192.168.1.41 web11

192.168.1.42 web22

192.168.1.43 db1

192.168.1.44 db2

192.168.1.45 cache

for i in web11 web22 db1 db2 cache

do

scp /etc/yum.repos.d/local.repo $i:/etc/yum.repos.d/local.repo

scp /etc/hosts $i:/etc/hosts

done

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

[root@ansible ~]#ansible --version //查看版本

**主机定义与分组 ad-hoc**

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

14 inventory = /etc/ansible/hosts

61 host\_key\_checking = False //远程时不需要输入yes

如果为False,不需要输入 yes

如果为 True,等待输入 yes

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

[web]

web11

web22

[db]

db[1:2] //db1到db2

[other]

cache

**ansible命令基础**

-m 模块名称,默认command模块

-a or --args 模块参数

-i inventory 文件路径,或可执行脚本

-k 使用交互式登录密码

-e 定义变量

-v 显示详细信息

[root@ansible ~]# ansible web --list-hosts //列出web里有多少主机

[root@ansible ~]# ansible all --list-hosts //列出所有主机

[root@ansible ~]# ansible web -m ping -k //批量检测主机 (会报错)

[root@ansible ~]#ssh-keygen -t rsa -b 2048 -N '' -f key

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

[root@ansible .ssh]# ssh-copy-id -i key.pub web11

[root@ansible .ssh]# ssh-copy-id -i key.pub web22

[root@ansible .ssh]# ssh-copy-id -i key.pub db1

[root@ansible .ssh]# ssh-copy-id -i key.pub db2

[root@ansible .ssh]# ssh-copy-id -i key.pub cache

[root@ansible .ssh]# ansible all -m ping //(全绿)

删除缓存

[root@ansible ~]#cd /root/.ansible/cp/

[root@ansible cp]# rm -rf

**inventory扩展参数**

更改端口: -ansible\_ssh\_port

更改ssh用户: -ansible\_ssh\_user

SSH密码: -ansible\_ssh\_pass

SSH使用的私钥文件 -\_\_ansible\_ssh\_private\_key\_file

vars 变量定义,用于组名后面

-[all:vars]

-ansible\_ssh\_private\_key\_file="/root/.ssh/key"

children 子组定义,用于引用其他组名称

-[app:children]

-web

-db

分组定义、范围定义样例

[root@db2 ~]# vim /etc/ssh/sshd\_config

Port 222

[root@db2 ~]# systemctl restart sshd

给所有主机部署key

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

[db]

db1

db2 ansible\_ssh\_port\_222

[other]

cache

[all:vars]

ansible\_ssh\_private\_key\_file="/root/.ssh/key"

[root@ansible .ssh]# ansible all -m ping

配置主机分组,自定义配置文件,重新定义一个新的ansible.cnf

[root@ansible myansible]#mkdir myansible

[root@ansible myansible]#cd myansible/

[root@ansible myansible]#vim myhost

[app1]

web11

db1

[app2]

web22

db2

[app:children]

app1

app2

[other]

cache

[all:vars]

ansible\_ssh\_private\_key\_file="/root/.ssh/key"

[root@ansible myansible]#touch ansible.cnf

[root@ansible myansible]# vim ansible.cnf

[defaults]

inventory = myhost

host\_key\_checking = Faise

**ansible-doc 和 ping 模块**

列出所有模块,并统计个数

ansible-doc -l | wc -l

ansible-doc modulename 查看帮助

ansible all -m ping 测试ssh的连通性

**command模块**

注意事项:

命令有以下字符则不成功 < > | &

command 模块不能解析系统变量

如果不使用shell模块,所有使用shell命令执行都不成功

查看db组的所有机器的 负载

[root@ansible ~]# ansible db -m command -a 'uptime'

db1 | SUCCESS | rc=0 >>

14:26:27 up 4:17, 1 user, load average: 0.00, 0.01, 0.03

db2 | SUCCESS | rc=0 >>

14:26:27 up 4:17, 2 users, load average: 0.00, 0.01, 0.03

ansible all -m command -a 'date +%F\_%T'

**SHELL模块**

用法基本和command一样,区别shell通过/bin/sh进行执行命令,可以执行任意命令,不能执行交互式的命令,列如: vim top等

使用shell执行命令

[root@ansible ~]# ansible web -m shell -a 'ps aux| grep ssh'

查看所有机器的负载

ansible all -m shell -a 'uptime'

转译

[root@ansible ~]# ansible web -m shell -a "echo \\\${HOSTNAME}"

[root@ansible ~]# ansible web -m shell -a "echo \\${HOSTNAME}"

[root@ansible ~]# ansible web -m shell -a "echo \${HOSTNAME}"

问题解析:

变量解析

ansible执行命令是二次解析

第一次在本机解析,第二次在执行机器解析

需要第二次解析的变量要转译 (\)

创建切换目录

ansible web -m shell -a "touch testfile"

ansible web -m shell -a 'chdir=/testfile touch ppp'

**script模块**

在本地写脚本,然后使用script模块批量执行

列:

ansible all -m script -a '/root/.ssh/a.sh'

**yum模块**

name:要进行操作的软件包名字

state: 动作(installed, removed)

install == installed

remove == removed

安装mariadb-server

ansible db -m yum -a 'name="mariadb-server" state=installed'

删除lrzsz

ansible cache -m yum -a 'name="lrzsz" state=remouved'

**service模块**

name:必选项,服务名称

enabled: 是否开机启动 yes | no

sleep:执行restared,会在stop 和start 之间沉睡几秒钟

state:对当前服务执行启动,停止,重启,重新加载等操作 (started **stopped** restarted reloaded)

ansible all -m server -a 'name="sshd" enabled="yes" state="started"'

**copy模块**

复制文件到远程主机

src 复制本地文件到远程主机,可以使,绝对或相对路径.

以"/"结尾,只复制目录里的内容,

不以"/"结尾,复制包含目录在内的整个内容,类似于rsync

dest 远程主机绝对路径,如果是目录,那么路径必须是目录

backup: 覆盖前先备份原文件,备份文件包含时间信息.有 yes | no 选项

force: 如果目标主机包含该文件,但内容不同,如果是 yes强制覆盖,no时,不存在则复制,默认yes

复制文件

ansible db -m copy -a 'src=/root/my.cnf dest=/etc/my.cnf'

复制目录

ansible db -m copy -a 'src=/etc/haha/ dest=/etc/haha**/**'

**lineinfile模块**

类似于sed的一种行编辑替换模块

path 目标文件

regexp 正则表达式,要修改的行

line 最终修改的结果

[root@ansible ~]# ansible db -m lineinfile -a 'path=/etc/my.cnf regexp="^binlog-format" line="binlog-format=row"

**replace模块**

类似于sed的一种行编辑替换模块

path 目标文件

regexp 正则表达式,

replace 替换后的结果

[root@ansible ~]# ansible db -m replace -a 'path=/etc/my.cnf regexp="row$" replace="mixed"'

**setup模块**

主要用于获取主机信息,playbooks里经常会用的另一个参数gather\_facts与该模块相关,setup模块下经常用的是filter参数

filter过滤所需信息

[root@ansible ~]# ansible cache -m setup -a 'filter=ansible\_distribution'