Ansible

It自动化和devops软件

基于Python研发

作用: 自动化批量管理主机

自动化, 批量

实质:

**就是个命令,没有后台服务,模块化设计**

代替人工ssh连接到主机执行命令

]# ansible 主机集合 -m 模块 -a “命令(模块参数)”

特点:

配置文件加载顺序,实现可有无数配置文件

**执行核心: 主机集合 命令集合**

运行原理:

读取配置-->

->分组列表-->过滤机器列表 //主机集合

->根据参数确定执行模块和配置 //命令集合

->runner执行返回

->输出,结束

要求:

必须开ssh和python2.6 - 2.7(都是系统自带)

配置文件(默认):

配置文件: /etc/ansible/ansible.cfg

主机文件: /etc/ansible/hosts

(可从静态文件中读取分组)

优点: {活跃度, 学习成本低, 使用成本低

使用是否广泛(是否有人趟雷bug等)}

只需ssh和python即可使用(都是系统自带)

无客户端

功能强大,模块丰富

上手容易,门槛低

Python优点: 编写相对简单, 放在任何机器都能用,系统自带python解释器,

(yum就是Python写的)

##################################################################################

ansible 主机集合 -m 模块 -a “命令(模块参数)”

]# ansible 组名|主机名 --list-hosts //查看主机

]# ansible 组名|主机名 -m ping -k //测试是否能通

]# ansible-doc -l //查看模块

]# ansible 主机|组 -m 模块 -a “命令” //运用模块

-------------------------------------------------------------------------------

Inventory参数

• vars 变量

[web]

主机名或ip ansible\_ssh\_port=222 //单独声明变量

[all:vars] //[对所有组:声明变量]

变量内容

• children 子组定义: 引用组的名称(解决重复书写)

[app:children] //查看app组会列出web组和db组

web //web组

db //db组

########################################################################################

自定义分组

(主机的集合)

通过配置文件加载顺序 实现自定义分组,可有无数组配置文件

首先检测: 变量 (ANSIBLE\_CONFIG变量定义的配置文件)

其次检查: 当前目录 (当前目录下的./ansible.cfg 文件)

再次检查: 家目录 (当前用户家目录下 ~/ansible.cfg 文件)

最后检查: 默认 (/etc/ansible/ansible.cfg 文件)

自己编辑配置文件的: 14 和 61 行

自己编辑hosts文件

####################################################################################

部署ansible

都为2cpu,1.5G以上内存,10G以上硬盘

1.40 ansible

1.41 web1

1.42 web2

1.43 db1

1.44 db2

---------------------------------------------------------------------------------------------------

1. 将04/ansible/ 下的所有包复制到ftp下做成yum源

[root@ansible ~]# createrepo --update

1. ansible主机配置新yum源
2. 下载ansible

[root@ansible ~]# yum install ansible

1. 修改配置文件

打开注释:

]# vim /etc/ansible/ansible.cfg

14 inventory = /etc/ansible/host

61 host\_key\_checking = False

//false: 第一次ssh自动yes

//true : 等待输入用户yes

加组及主机:

]# vim /etc/ansible/hosts

[组名称]

主机名或ip

主机名或ip ansible\_ssh\_port=222 //单独声明变量

[all:vars] //所有组声明变量

ansible\_ssh\_private\_key\_file=”密钥路径”

1. 配置密钥

]#ssh-keygen -t rsa -b 2048 -N ‘’ -f key

]#ssh-copy-id -i key.pub 主机名

]# ansible 组名|主机名 --list-hosts //查看主机

all: 所有主机

逗号分隔多个参数

]# ansible 组名|主机名 -m ping -k //检测各主机是否能通(-k: 交互输入密码)

###############################################################################

**模 块**

**总命令:**

]# ansible-doc -l //查看模块

]# ansible 主机|组 -m 模块 -a “命令”

**各模块功能**

**Command模块(默认)**

– 远程不启动bash,有很多问题

– 因为没有启动bash,所有不能用重定向,管道

]# ansible db -m command -a 'uptime'

**Shell模块**

– 远程启动shell

**\*\* 一个大坑 \*\* :**

两次解析: 本机bash先解析,再给ansible

]# ansible db -m shell -a **“**echo ${HOSTNAME}**”**

ansible

]# ansible db -m shell -a **‘**echo ${HOSTNAME}**’**

db

db1

练习

给web1,db2添加用户nb,并设置密码为123

]# ansible web1,db2 -m shell -a "useradd nb;echo 123 | passwd --stdin nb"

**Script模块**

– 批量执行脚本

将本机的脚本批量发给主机执行

]#ansible db -m script -a ‘脚本’

**Yum 安装模块**

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

**Service 启动服务模块**

]# ansible db -m service -a 'name=mariadb state=started enabled=yes'

**Copy复制模块**

src: 源路径

dest: 目标路径

**Lineinfile (烂因file) 模块**

– 整行修改(像sed)

[root@ansible ~]# ansible db -m lineinfile -a ' \

> path="/etc/my.cnf" \ //修改的文件

> regexp="^binlog-format" \ //文件里的哪一行

> line="binlog-format = row"' //修改后的内容

**Replace(瑞pe里死) 模块**

– 只修改匹配的部分(更像sed)

[root@ansible ~]# ansible db -m replace -a ' \

path="/etc/my.cnf" \

regexp="row$" \

replace="wrow"' //不是line

**setup模块**

**查看主机信息**