ansible自动化运维

人工运维时代

运维人员早期需要维护数量众多的机器，因此需要执行反复，重复的劳动力，很多机器需要同时部署相同的服务或是执行相同的命令，还得反复的登录不同的机器，执行重复的动作。。

比如说你要在backup服务器配置rsync服务，进行数据同步的操作，那么客户端都得单独的安装一下rsync命令工具才能正确使用

你可能一台台机器去登录，安装rsync之后，再推出，登录下一个机器，可以使用xshell等工具，快捷的创建ssh登录，但是还是属于人工运维，效率比较低

自动化运维时代

早期运维人员会结合ssh免密登录以及shell脚本来完成自动化的部署操作。

系统管理员面临的问题主要是，配置管理系统，远程执行命令，批量安装服务，启停服务等等

后来也就诞生了众多的开源软件，自动化运维软件

- fabric
- puppet
- saltstack
- chef
- Ansible <<<<<<<<<<

其中有两款软件是基于python语言开发的，saltstack,ansible都是基于python编写

自动化运维趋势

人肉运维，人力运维 > 自动化运维 > 数据化运维，可视化运维 > AI智能运维 ，devops

自动化运维的好处

- 提高工作效率，减少重复性的劳动力操作
- 大大的减少了人为出错的可能性
- ansible支持数据化管理，数据化追源，找到问题的来源点

ansible介绍

ansible是一个同时管理多个远程主机的软件，必须是任意可以通过ssh登录的机器，因此ansible可以管理的机器如

- 远程虚拟机
- 物理机
- 也可以直接管理本机机器

ansible通过shh协议实现了，管理节点(老板,安装了ansible服务的机器)，被管理节点（员工，被管理的机器节点）的通信。

只能是通过ssh协议登录的主机，就可以完成ansible自动化部署操作

- 批量文件分发
- 批量数据复制
- 批量数据修改，删除
- 批量自动化安装软件服务
- 批量服务启停
- 脚本化，自动批量服务部署

ansible特点

ansible的编排引擎可以出色的完成各种任务配置管理，ansible在流程控制，资源部署等方便很强大，并且ansible无须安装客户端软件，管理简洁，使用yaml配置文件语法，功能强大，便于维护。

ansible是基于python语言开发的，主要由python的两个ssh处理模块，paramiko，以及PyYAML模块

- 安装部署简单
- 管理主机便捷，支持多台主机并行管理
- 无须安装被管理节点的客户端(no agent)，且无须占用客户端的其他端口，仅仅使用ssh服务即可
- 不仅仅支持python，还支持其他语言的二次开发
- 不用root用户也可执行，降低系统权限

Ansible实践部署

准备好虚拟机

```
准备好3个linux虚拟机，配置在同一个局域网内，然后设置好静态ip地址

rsync01    192.168.178.139        被管理机器
nfs01      192.168.178.138        被管理机器（配置好ssh服务，以及关闭防火墙等等，公司的员工）
m01        192.168.178.120        管理机器（安装了ansible的服务端，公司的老板）
```

先准备ansible管理机器(老板)

```
1.选择yum自动化安装，阿里云yum，epel源，前提就得配置好
yum install  epel-release -y
yum install   ansible   libselinux-python -y

2.检查ansible软件安装情况，查询配置文件，和可执行命令

[root@m01 ~]# rpm -ql ansible | grep -E '^/etc|^/usr/bin'

3.检查ansible版本
ansible --version
```

再准备ansible被管理机器（员工）

安装ansible所需的系统模块

```
nfs01
rsync01

yum install epel-release  libselinux-python -y
```

ansible管理方式

ansible批量管理主机的方式主要有两种

- 传统的输入ssh密码验证
- 密钥管理

```
配置好ansible的配置文件，添加被管理机器的ip地址，或者主机名
1.备份现有的配置文件
cp /etc/ansible/hosts{,.ori}

2.添加ansible需要管理的机器地址，添加如下信息
[root@m01 ansible]# tail -3 /etc/ansible/hosts
[chaoge]
192.168.178.138
192.168.178.139
```

ssh密码认证方式管理机器

ansible是直接利用linux本地的ssh服务，以及一些远程的ssh操作，一般情况下客户端的ssh服务默认都是开启的，无须额外管理

```
1.在m01机器上执行如下命令
-m 指定功能模块，默认就是command模块
-a 告诉模块需要执行的参数
-k 询问密码验证
-u 指定运行的用户
在m01机器上，告诉其他被管理的机器，你要执行什么命令，以及用什么用户去执行
ansible  chaoge   -m   command  -a 'hostname'  -k -u root

2.如上操作，一般默认情况下回有错误提示，需要执行如下动作，只需要手动ssh对主机进行一次连接，即可使用ansible命令操作了
ssh root@192.168.178.138
ssh root@192.168.178.139


3.此时可以再次执行ansible命令，在m01管理机器上，让2个被管理的机器，执行我们想要的结果
ansible chaoge -m command -a "ifconfig ens33"  -k -u root
```

配置免密登录

每次执行ansible命令的时候，都需要输入ssh的认证密码，也就是root的密码，如果不同的主机密码不一致，那你还得输入多次才行

因此我们可以配置如下的快捷登录方式

ansible自带的密码认证参数

```
可以在 /etc/ansible/hosts文件中，定义好密码即可，即可实现快速的认证，远程管理主机

参数
ansible_host       主机地址
ansible_port       端口,默认是22端口
ansible_user       认证的用户
ansible_ssh_pass   用户认证的密码
```

使用hosts文件的参数形式，来实现ssh认证

```
1.修改hosts文件，改为如下
[chaoge]
192.168.178.138 ansible_user=root  ansible_ssh_pass=111111
192.168.178.139 ansible_user=root  ansible_ssh_pass=111111

2.此时可以不需要输入密码，即可自动ssh验证通过了
ansible chaoge -m command -a "ifconfig ens33"
```

ssh密钥方式批量管理主机

这个方式比起hosts文件的密码参数来的更安全放心

```
1.在m01机器上创建ssh密钥对
ssh-keygen -f ~/.ssh/id_rsa -P ""  > /dev/null 2>&1

2.此时检查公私钥文件
[root@m01 ansible]# cd ~/.ssh/
[root@m01 .ssh]# ls
authorized_keys  id_rsa  id_rsa.pub  known_hosts
```

编写公钥分发脚本

```
#!/bin/bash
rm -rf ~/.ssh/id_rsa*
ssh-keygen -f ~/.ssh/id_rsa -P "" > /dev/null 2>&1
SSH_Pass=111111
Key_Path=~/.ssh/id_rsa.pub
for ip in 138 139
do
        sshpass -p$SSH_Pass ssh-copy-id -i $Key_Path "-o StrictHostKeyChecking=no" 192.168.178.$ip
done
# 非交互式分发公钥命令需要用sshpass指定SSH密码，通过-o StrictHostKeyChecking=no 跳过SSH连接确认信息


```

此时在m01机器上再连接客户端机器，就无须输入账号密码了，可以尝试使用ansible命令进行连接

```
ansible  chaoge -m command -a "uname -a"
```

此时已经无须输入密码，即可远程管理

总结

在生产环境中，ansible的连接方式，二选一即可，最好的是配置ssh公私钥免密登录

如果生产环境的要求更高，可以用普通用户去执行，再提权操作，sudo

ansible模式与命令

ansible实现批量化主机管理的模式，主要有两种

- 利用ansible的纯命令行实现的批量管理，ad-hoc模式---------好比简单的shell命令管理
- 利用ansible的playbook剧本来实现批量管理，playbook剧本模式------好比复杂的shell脚本管理

ad-hoc模式

ansible的ad-hoc模式是ansible的命令行形式，也就是处理一些临时的，简单的任务，可以直接使用ansible的命令行来操作

比如

- 临时批量查看被管理机器的内存情况，cpu负载情况，网络情况
- 比如临时的分发配置文件等等

playbook模式

ansible的playbook模式是针对比较具体，且比较大的任务，那么你就得实现写好剧本，应用场景

- 一键部署rsync备份服务器
- 一键部署lnmp环境

ansible的ad-hoc命令行解析

```
m01         ansible管理机器------老板
rsync01     被管理机器-----员工
nfs01       被管理机器-----员工
```

让被管理机器返回主机名

```
[root@m01 ~]# ansible  chaoge  -m command  -a "hostname"
192.168.178.138 | CHANGED | rc=0 >>
nfs01
192.168.178.139 | CHANGED | rc=0 >>
rsync01
```

ad-hoc命令解释

- Ansible ---自带提供的命令操作
- Chaoge------/etc/ansible/hosts文件中定义的主机组，还可以写主机ip地址，以及通配符*
- -m command ansible的指定模块的参数，以及指定了command模块
- -a 指定给command模块什么参数，hostname，uname -r

ansible-doc命令

```
列出所有的ansible支持的模块
ansible-doc -l |grep ^command

查看某个模块的具体用法参数
ansible-doc  -s command
```

Ansible模块精讲

Ansible-doc -l

command模块

作用：在远程节点上执行一个命令

ansible-doc -s command 查看该模块支持的参数

```
chdir    在执行命令之前，先通过cd进入该参数指定的目录
creates    在创建一个文件之前，判断该文件是否存在，如果存在了则跳过前面的东西，如果不存在则执行前面的动作
free_form    该参数可以输入任何的系统命令，实现远程执行和管理
removes    定义一个文件是否存在，如果存在了则执行前面的动作，如果不存在则跳过动作
```

command模块是ansible的默认基本模块，也可以省略不写，但是要注意如下的坑

- 使用command模块，不得出现shell变量 `$name` ，也不得出现特殊符号 `> < | ; &` 这些符号command模块都不认识，如果你想用前面指定的变量，特殊符号，请使用 `shell模块` ，command模块就不适合你了

command模块案例

获取所有被管理机器的负载信息

```
ansible  chaoge  -m command -a "uptime"

[root@m01 ~]# ansible  chaoge  -m command -a "uptime"
192.168.178.138 | CHANGED | rc=0 >>
 11:38:39 up 56 min,  2 users,  load average: 0.06, 0.04, 0.05
192.168.178.139 | CHANGED | rc=0 >>
 11:38:39 up 40 min,  2 users,  load average: 0.00, 0.01, 0.05
```

让客户端机器，先切换到/tmp目录下，然后打印当前的工作目录

```
ansible chaoge -m command -a "pwd"

[root@m01 ~]# ansible chaoge -m command -a "pwd chdir=/tmp/"
192.168.178.138 | CHANGED | rc=0 >>
/tmp
192.168.178.139 | CHANGED | rc=0 >>
/tmp
```

练习creates参数

该参数作用是判断该文件是否存在，存在则跳过，不存在则执行

```
#判断/chaoge文件夹是否存在，存在则不执行前面的pwd动作，不存在，则执行pwd
ansible  192.168.178.139  -m command -a "pwd  creates=/chaoge"

[root@m01 ~]# ansible  192.168.178.139  -m command -a "pwd creates=/opt"
192.168.178.139 | SUCCESS | rc=0 >>
skipped, since /opt exists
```

参数removes实践，存在则执行，不存在则跳过

```
ansible chaoge -a "ls /opt     removes=/chaoge"

  ansible chaoge -a "ls /opt     removes=/chaoge"
  ansible chaoge -a "ls /op"
  ansible chaoge -a "ls /op  removes=/op"
  ansible chaoge -a "ls /opt  removes=/opt"
```

warn参数，是否提供告警信息

```
# 执行命令，且忽略告警信息
[root@m01 ~]# ansible  chaoge -m command -a "chmod 000 /etc/hosts   warn=False"
192.168.178.139 | CHANGED | rc=0 >>

192.168.178.138 | CHANGED | rc=0 >>
```

shell模块

作用：在远程机器上执行命令（复杂的命令）

了解模块用法的渠道

- linux命令行里面通过 ansible-doc
- ansible官网查看帮助信息 https://docs.ansible.com/ansible/latest/modules/shell_module.html

```
shell模块支持的参数和解释

chdir      在执行命令之前，通过cd进入该参数指定的目录
creates    定义一个文件是否存在，如果存在则不执行该命令，如果存在该文件，则执行shell命令
free_form  参数信息中可以输入任何的系统指令，实现远程管理
removes    定义一个文件是否存在，如果存在该文件，则执行命令，如果不存在，则跳过
```

shell模块案例

```
批量查询进程信息

[root@m01 ~]# ansible chaoge -m shell  -a "ps -ef|grep vim|grep -v grep"
192.168.178.138 | CHANGED | rc=0 >>
root       4762   4741  0 14:39 pts/0    00:00:00 vim xixixi.sh
192.168.178.139 | CHANGED | rc=0 >>
root       1160   1145  0 14:38 pts/0    00:00:00 vim heiheihei.txt

```

批量在客户端机器，创建写入文件信息

```
[root@m01 ~]# ansible chaoge -m shell -a "echo 你真棒 > /tmp/heihei.txt"
192.168.178.139 | CHANGED | rc=0 >>

192.168.178.138 | CHANGED | rc=0 >>
```

批量远程执行脚本

```
该需要执行的脚本，必须要求在客户端机器上存在，否则会报错文件不存在，这是shell模块的特点，是因为还有一个专门执行脚本的script模块
注意的是这个脚本必须在客户端机器上存在才行
1.创建文件夹
2.创建sh脚本文件，还要写入脚本内容
3.赋予脚本可执行权限
4.执行脚本
5.忽略warning信息
```

```
思路分析
最好所有的操作都是在 管理机器上，也就是(老板)这台机器 m01上进行远程的，批量化操作

ansible  chaoge -m shell -a "mkdir -p /server/myscripts/;echo 'hostname' > /server/myscripts/hostname.sh;chmod +x /server/myscripts/hostname.sh;bash /server/myscripts/hostname.sh  warn=False"
```

script模块

功能：吧m01管理机器上的脚本远程的传输到备管理节点上去执行

比起shell模块，script模块功能更强大，在m01机器本地有一份脚本，就可以在所有被管理节点上去运行

script的模块参数

```
creates
removes
chdir
```

应用案例

```
1.在管理节点上创建脚本
[root@m01 myscripts]# echo -e "pwd\nhostname" > /myscripts/local_hostname.sh
[root@m01 myscripts]#
[root@m01 myscripts]#
[root@m01 myscripts]# cat /myscripts/local_hostname.sh
pwd
hostname

2.授权
chmod +x  /myscripts/local_hostname.sh
```

远程的批量执行脚本，且在客户端上不需要存在该脚本

```
ansible  chaoge -m script -a "/myscripts/local_hostname.sh"
```

利用script模块 可以批量让所有被管理的机器执行脚本，且该脚本不需要在客户端上存在

ansible文件操作的模块

copy模块

作用：复制文件数据到远程主机

```
ansible-doc -s copy  #查看copy模块的参数用法

参数解释如下

```

copy模块是远程推送数据的模块，只能把管理节点上的数据，推送给远程节点，无法拉取数据到本地

实际案例

```
1.吧m01上的文件数据，发给被管理节点

2.先创建好需要数据复制的 user group，批量创建用户 用户组，通过command模块或者shell模块，远程的执行命令即可
[root@m01 ~]# ansible chaoge -m command -a "useradd learn_ansible"
192.168.178.139 | CHANGED | rc=0 >>

192.168.178.138 | CHANGED | rc=0 >>

[root@m01 ~]# ansible chaoge -m command -a "id learn_ansible"
192.168.178.139 | CHANGED | rc=0 >>
uid=8890(learn_ansible) gid=8890(learn_ansible) 组=8890(learn_ansible)
192.168.178.138 | CHANGED | rc=0 >>
uid=8889(learn_ansible) gid=8889(learn_ansible) 组=8889(learn_ansible)


3.批量拷贝文件，分发给客户端节点
ansible  chaoge -m copy  -a "src=/etc/hosts dest=/tmp/m01_hosts  owner=learn_ansible group=learn_ansible mode=0666"

远程检查拷贝后的文件信息
[root@m01 ~]# ansible chaoge -m command -a "ls -l /tmp/m01_hosts"
192.168.178.139 | CHANGED | rc=0 >>
```

```
-rw-rw-rw- 1 learn_ansible learn_ansible 158 5月  27 16:57 /tmp/m01_hosts
192.168.178.138 | CHANGED | rc=0 >>
-rw-rw-rw- 1 learn_ansible learn_ansible 158 5月  27 16:57 /tmp/m01_hosts
```

远程批量复制文件，备份，追加内容

```
1.批量远程的生成文件和内容
ansible chaoge -m shell -a "echo 今天天气不错 > /tmp/day.txt"

验证文件内容
[root@m01 ~]# ansible chaoge -m shell -a "cat /tmp/day.txt"
192.168.178.139 | CHANGED | rc=0 >>
今天天气不错
192.168.178.138 | CHANGED | rc=0 >>
今天天气不错


2.就是批量的实现了文件远程拷贝，且定义了新的内容放入文件中，并且真对目标机器的源数据文件，做了一个备份
ansible chaoge -m  copy -a "content='Hello,my name is chaoge,who are u' dest=/tmp/day.txt backup=yes"

远程批量验证文件结果
[root@m01 ~]# ansible chaoge -m shell -a "ls -l /tmp/day*"
192.168.178.138 | CHANGED | rc=0 >>
-rw-r--r-- 1 root root 33 5月  27 17:20 /tmp/day.txt    # 这是新生成的文件，和内容
-rw-r--r-- 1 root root 19 5月  27 17:18 /tmp/day.txt.2096.2020-05-27@17:20:39~   #这是backup参数做好的文件备份
192.168.178.139 | CHANGED | rc=0 >>
```

```
-rw-r--r-- 1 root root 33 5月  27 17:20 /tmp/day.txt
-rw-r--r-- 1 root root 19 5月  27 17:18 /tmp/day.txt.1879.2020-05-27@17:20:39~


```

file模块

作用，创建，修改文件，目录的属性

```
ansible-doc -s file  #查看模块详细信息与用法

file模块常用的参数解释:
group  定义文件/目录的 属组
owner   定义属主
mode   定义权限
path  必选参数，定义文件路径
src    定义源文件路径，主要用于创建link类型文件使用
dest  创建出来的软连接 它的路径
state 参数:
  file  : 如果目标文件不存在，那么不会创建该文件
  touch:  如果文件不存在，则创建一个新的文件，如果文件已经存在了，则修它的最后修改时间
  directory: 如果目录不存在，那么会创建目录
  link:  用于创建软连接类型
  absent : 删除目录，文件或者取消连接

```

fiile模块主要用于创建文件，目录，以及文件数据，或者对现有的文件，目录修改权限

file实践

```
1.远程的批量创建文件夹，并且设置权限是666
[root@m01 ~]# ansible  chaoge -m file  -a "dest=/tmp/cc_dir/ mode=666 state=directory"

2.验证文件夹是否存在，以及权限查看
ansible chaoge -m shell -a "ls -ld /tmp/cc_dir"
```

远程批量生成文件

```
# 目标文件不存在，则不执行动作，这是state的file属性
ansible  chaoge -m file -a "dest=/tmp/cc_666.txt state=file owner=chaoge group=chaoge mode=600"

#应该使用state的touch属性
ansible  chaoge -m file -a "dest=/tmp/cc_666.txt state=touch owner=chaoge group=chaoge mode=600"

[root@m01 ~]# ansible chaoge -m shell -a "ls -l /tmp/cc_666.txt"
192.168.178.139 | CHANGED | rc=0 >>
-rw------- 1 chaoge chaoge 0 5月  28 11:12 /tmp/cc_666.txt
192.168.178.138 | CHANGED | rc=0 >>
-rw------- 1 chaoge chaoge 0 5月  28 11:12 /tmp/cc_666.txt
```

远程创建软连接

源文件绝对路径 软连接绝对路径

```
ansible  chaoge -m file -a "src=/etc/hosts
dest=/tmp/ansible_hosts_test state=link"

#验证该软连接
[root@m01 ~]# ansible chaoge -m shell -a "ls -l
/tmp/ansible_hosts_test"
192.168.178.139 | CHANGED | rc=0 >>
lrwxrwxrwx 1 root root 10 5月  28 11:17 /tmp/ansible_hosts_test
-> /etc/hosts
192.168.178.138 | CHANGED | rc=0 >>
lrwxrwxrwx 1 root root 10 5月  28 11:17 /tmp/ansible_hosts_test
-> /etc/hosts
```

yum模块练习

实践1

```
1.批量检查所有被管理节点是否安装了nginx服务
[root@m01 ~]# ansible chaoge -m shell -a "rpm -qa nginx
warn=false"
192.168.178.139 | CHANGED | rc=0 >>

192.168.178.138 | CHANGED | rc=0 >>

2.通过yum模块批量安装服务
ansible chaoge -m yum  -a "name=nginx state=installed"

3.远程的检查服务是否安装了
[root@m01 ~]# ansible chaoge -m shell -a "rpm -qa nginx
warn=false"
192.168.178.139 | CHANGED | rc=0 >>
nginx-1.18.0-1.el7.ngx.x86_64
```

```
192.168.178.138 | CHANGED | rc=0 >>
nginx-1.18.0-1.el7.ngx.x86_64

4.批量远程卸载nginx
[root@m01 ~]# ansible chaoge -m yum -a "name=nginx
state=absent"

5.此时再次检查nginx是否被卸载
[root@m01 ~]# ansible chaoge -m shell -a "rpm -qa nginx"
[WARNING]: Consider using the yum, dnf or zypper module rather
than running 'rpm'.  If you need to use
command because yum, dnf or zypper is insufficient you can add
'warn: false' to this command task or set
'command_warnings=False' in ansible.cfg to get rid of this
message.
192.168.178.139 | CHANGED | rc=0 >>

192.168.178.138 | CHANGED | rc=0 >>


6.升级软件包，指定升级nginx，也可以写成 name='*' 就等于 yum
update升级所有软件包，latest也提供下载更新
ansible chaoge -m yum -a "name='nginx' state=latest"

[root@m01 ~]# ansible chaoge -m shell -a "rpm -qa nginx"
[WARNING]: Consider using the yum, dnf or zypper module rather
than running 'rpm'.  If you need to use
command because yum, dnf or zypper is insufficient you can add
'warn: false' to this command task or set
'command_warnings=False' in ansible.cfg to get rid of this
message.
192.168.178.139 | CHANGED | rc=0 >>
nginx-1.18.0-1.el7.ngx.x86_64
192.168.178.138 | CHANGED | rc=0 >>
```

```
nginx-1.18.0-1.el7.ngx.x86_64


7.升级系统所有软件包，排除某个服务不升级，这个命令，注意不要在服务器上随便敲，因为服务器不得任意更新一些服务版本，可能会造成服务挂掉
ansible chaoge -m yum -a "state=latest  name='*' exclude='nginx'"


```

Ansible服务管理模块

通过yum命令安装的软件，在centos6和centos7平台下有不同的启动命令

Centos6----serivce

```
service  nginx start/stop/restart/reload
```

Centos7--systemctl

```
systemctl start/stop/restart/reload/status/     nginx.serivce
```

通过ansible的yum模块安装的软件，我们还可以通过远程批量化的服务管理模块，进行批量的启停

针对service命令，用在centos6系统平台上

针对systemctl命令，主要用于centos7平台

serivce/systemd模块

```
 ansible-doc -s service
 ansible-doc -s systemd
 要注意的是serivce已然对centos7有效
 当你使用service命令管理服务，系统自动的重定向为systemctl服务管理命令

 systemd模块
 name  指定服务的名字，比如nginx.serivce   如 crond.serivce
 state  填入你要执行的操作，如 reloaded,restarted,started,stopped
 enabled  指定服务开机自启  systemctl enable nginx
 daemon_reload  每当修改了配置文件，使用systemd重读配置文件
```

管理crond服务

```
1.远程的查看crond服务是否正常
[root@m01 ~]# ansible chaoge -m shell -a "systemctl status crond" |grep Active
   Active: active (running) since 五 2020-05-29 10:12:15 CST; 1h 7min ago
   Active: active (running) since 五 2020-05-29 10:12:11 CST; 1h 7min ago

2.检查crond服务是否开机自启了
[root@m01 ~]# ansible chaoge -m shell -a "systemctl list-unit-files" |grep crond
crond.service                                 enabled
crond.service                                 enabled

3.通过systemd模块管理服务
ansible chaoge -m systemd -a "name=crond  state=stopped"
ansible chaoge -m systemd -a "name=crond  state=started"
ansible chaoge -m systemd -a "name=crond  state=restarted"
```

```
15 ansible chaoge -m systemd -a "name=crond  state=reloaded"
```

管理nginx服务

```
1.检查客户端机器，nginx是否安装了
[root@m01 ~]# ansible chaoge -m shell -a "rpm -qa nginx"
[WARNING]: Consider using the yum, dnf or zypper module rather
than running
'rpm'.  If you need to use command because yum, dnf or zypper
is insufficient you
can add 'warn: false' to this command task or set
'command_warnings=False' in
ansible.cfg to get rid of this message.
192.168.178.139 | CHANGED | rc=0 >>
nginx-1.18.0-1.el7.ngx.x86_64
192.168.178.138 | CHANGED | rc=0 >>
nginx-1.18.0-1.el7.ngx.x86_64


2.启动nginx服务
ansible chaoge -m systemd  -a "name=nginx state=started
enabled=yes"
ansible chaoge -m systemd  -a "name=nginx state=stopped
enabled=no"
ansible chaoge -m systemd  -a "name=nginx state=restated"
ansible chaoge -m systemd  -a "name=nginx state=reloaded"
```

cron模块

crond服务，定时任务服务

Ansible---cron模块

cron模块主要是管理linux的定时任务条目

```
定时crontab条目都是遵循了规则

分 时  日 月 周   执行命令的绝对路径
*  *   *  *  *
*/5 * *  * *     没5分钟执行命令

每个月的3号，13号，早上8点整 重启nginx

0  8  3,13  *  *  /usr/bin/systemctl restart nginx

```

ansible的cron模块来添加任务

```
1.添加定时任务，每5分钟进行时间同步
ansible chaoge -m cron  -a "name=ntp_cron
job='/usr/sbin/ntpdate ntp.aliyun.com > /dev/null 2>&1'
minute=*/5 "

2.远程的查看定时任务是否添加
[root@m01 ~]# ansible chaoge -m shell -a "crontab -l"
192.168.178.138 | CHANGED | rc=0 >>
#Ansible: ntp_cron
*/5 * * * * /usr/sbin/ntpdate ntp.aliyun.com > /dev/null 2>&1
192.168.178.139 | CHANGED | rc=0 >>
0 0  *  *  *  /bin/bash   /myscripts/cut_nginx_log.sh
#Ansible: ntp_cron
*/5 * * * * /usr/sbin/ntpdate ntp.aliyun.com > /dev/null 2>&1

3.再添加一个记录，事件是每个月的3号，13号，早上8点整 重启nginx
思路：转化如下任务即可

```

```
0  8  3,13  *  *  /usr/bin/systemctl restart nginx

ansible  chaoge -m cron -a "name=restart_nginx
job='/usr/bin/systemctl restart nginx' minute=0 hour=8
day=3,13  "

4.删除定时任务，只能删除通过ansible模块添加的任务记录

ansible  chaoge -m cron -a "name='restart_nginx' state=absent"
```

ansible剧本

ansible核心的功能，作用就是进行配置管理

剧本

ansible需要编写的playbook剧本需要遵循一定的规则，格式，这个格式就称之为yaml语法

学习一下yaml语法

使用剧本批量安装nginx服务

```
1.批量卸载nginx服务

先查询被管理节点的机器，是否装了nginx
ansible chaoge -m shell -a "rpm -qa nginx"

批量卸载nginx
 ansible chaoge -m yum -a "name=nginx state=absent"
```

```
8
9  2.编写一个yaml配置文件，注意语法格式
10
11 mkdir /myyaml
12 编写你的第一个yaml配置文件，注意缩进的对其，和空格的数量
13 [root@m01 myyaml]# cat -n install_nginx.yaml
14      1  # install nginx yaml,by chaogo
15      2  - hosts: all
16      3    tasks:
17      4        - name: Install nginx service
18      5          yum: name=nginx state=present
19      6        - name: Copy Nignx.conf to every_server
20      7          copy: src=./nginx.conf
   dest=/etc/nginx/conf/nginx.conf mode=0644
21
22
```

解释上述的yaml语法

```
1 1.表示注释信息，可以用#号，也可以用三个短横线 当做注释
2 2.表示定义palybook管理的目标机器，可以写all管理所有，也可以单独的填写ip，也可以填写主机名
3 3.定义palybook需要完成的任务的集合，比如你要定义2，第一个是安装nginx通过yum模块，第二个是通过copy模块，发送管理节点上的配置文件，发送给被管理的客户端机器
4 4.定义任务的名字，是一个自定义的帮助信息
5 5.定义任务的具体操作，指定用哪些模块，与参数
6 6~7行作用是和4~5行是一样了
```

playbook剧本的组成规范

剧本很重要的就是，定义演员的信息(其实就是定义主机的信息)，演员的具体任务（以及主机要执行的模块，动作）

ansible的剧本也是由两个最基本的部分组成

- hosts定义的被管理的主机列表信息（演员有哪些）
- tasks关键词定义的被管理主机需要执行的动作（演员要做什么事）

剧本之hosts部分

```
# 方式一，定义被管理主机的ip地址
- hosts: 192.168.178.138
  tasks:
    - name: 这是我第一个任务
      yum: name=nginx  state=installed


# 方式二，定义被管理主机的名字，注意该主机名必须能够解析
- hosts: backup01
  tasks:
    - name: 需要执行的动作

# 方式三，定义多个主机信息
- hosts: 192.168.178.138,192.168.178.139,backup01
  tasks:
    - name: 执行的动作...

# 方式四，填写所有的主机
- hosts: all
  tasks:
```

定义被管理的主机，有一个重要的前提，就是被管理的主机，必须在

ansible管理的hosts文件中有对应的信息，否则识别不了

/etc/ansible/hosts

剧本之tasks讲解

- 变量形式定义task任务，name="超哥" age=18
- 还可以使用字典形式定义， 特点是 `key:value` 自，相关的词语，自卑，自豪，自律，自力更生
  - name:"超哥"
  - age:18
  - addr:"北京沙河"

```
#定义task方式一：采用变量形式设置任务
tasks:
  - name: make sure apache is runing
    service: name=httpd state=running

  - name: copy file..
    copy: src=/etc/ansible/hosts  dest=/etc/ansible/hosts
          owner=root group=root mode=0644


#当传入的参数列表过长的时候，我们还可以将其分割


# 方式二，采用字典形式设置多个任务
tasks:
  - name: copy file to client...
    copy:
      src: /etc/ansible/hosts
```

```
        dest: /etc/ansible/hosts
        owner: root
        group: root
        mode: 0644
```

yaml语法规范

在学习ansible 的时候，编写palybook是最重要的环节，那么palybook是遵循yaml语法，因此需要掌握yaml语法的数据格式

```
目前主流使用的数据格式
有json   xml    yaml   都属于数据序列化格式
yaml更容易被解析，已读，因此更多的使用在了配置文件当中
anslble   saltstack    k8s

yaml基本语法
   大小写敏感
   使用缩进表示层级的关系（同样的空格数量）
   在配置缩进关系的时候，禁止用tab键，请一个一个用空格键表示
   相同元素的左侧空格数目不重要，只需要对其即可

yaml语法支持的数据结构
   对象，指的就是   字典的概念 ，key: value    ，注意key:后面有一个空格
   数组，     指的是列表的概念
```

列表是什么?

例如

6期linux学员的名单

- 小张

- 小王
- 小李
- 小明

```
上述的关系可以用  yaml的数组形式表示

例如

"6期linux学员的名单":
  - "小敏"
  - "小李"
  - "小王"
  - "小于"
```

playbook编写的规范

剧本的缩进关系，一般是两个空格作为一个缩进，且空格数目无所谓，左侧对其即可

```
- hosts: chaoge
  tasks:
    - name: 安装nginx
      yum: name=nginx  state=installed
    - name: 执行脚本
      script: /server/scripts/test_ansible.sh
```

playbook实际编写与执行

剧本编写完成之后，还得执行才能工作

在ansible程序中，加载模块的功能可以直接使用ansible adhoc命令行形式执行

加载剧本中的功能，可以使用ansible-playbook命令去执行脚本

```
基本执行语法
ansible-playbook  nginx.yaml

可以使用绝对相对路径
```

查看剧本命令的帮相信息

```
ansible-playbook -h
```

查看剧本的执行详细输出

```
ansible-playbook  nginx.yaml  --verbose
```

查看剧本影响的主机列表信息

```
ansible-playbook nginx.yaml  --list-hosts
```

执行剧本加载指定的主机清单文件，默认剧本使用的是 `/etc/ansible/hosts`

```
ansible-playbook  nginx.yaml  -i  /etc/my_ansible/hosts
```

执行剧本并且检查语法

```
ansible-playbook  nginx.yaml  --syntax-check
```

调试剧本，只是调试，但是不会对被管理节点发生改变

```
ansible-playbook nginx.yaml -C  # 模拟执行，不影响客户端机器
```

实践执行剧本

playbook之部署rsync服务

```
1.进行思考，剧本如何设计，和安排

2.先考虑好部署rsync的整个过程

部署的思路过程如下，然后转化成playbook剧本的过程就可以
  1.安装rysnc服务，使用yum
  2.编写rsync配置文件，（常用的操作是，在m01管理机器上编写好配置文件，分发给客户端机器即可）copy模块
  3.创建rsync用户，使用ansible的user模块
  4.创建用于rsync验证的文件，还是选择在m01上写好后分发，copy模块，注意文件权限设置
  5.创建数据备份的文件夹，rsync指定了bakcup模块，备份目录是/data_backup/，调用file模块创建文件夹
  6.启动rsync服务端，且是守护进程模式，shell模块，systemd模块

```

上述部署思路转化为ansible playbook的写法

```
[root@m01 myyaml]# cat  install_rsync.yaml
- hosts: 192.168.178.115
  tasks:
    - name: step01,install rsync service
      yum: name=rsync state=installed

    - name: step02,edit rsync conf file


```

```
      copy: src=/etc/ansible/rsync_conf/rsyncd.conf dest=/etc/rsync/conf/

    - name: step03,create user rsync
      user: name=rsync state=present  createhome=no shell=/sbin/nolgoin

    - name: step04,create user auth file
      copy: src=/etc/ansible/rsync_conf/rsync.password dest=/etc/rsync/conf/ mode=0600

    - name: step05,create backup dir
      file: dest=/data_backup/  state=directory owner=rsync group=rsync

    - name: step06,run rsync server
      shell: rsync --daemon creates=/var/run/rsync.pid
```

运行剧本的方法

```
ansible-playbook install_rsync.yaml  -C
ansible-playbook install_rsync.yaml
```

留个作业，自行的一键部署rsync客户端的操作，以及思考如何用ansible的剧本，一键部署实时同步