# Ansible自动运维企业实战

随着互联网IT运维飞速发展，目前市场上涌现了大量的自动化配置维护工具，例如 PSSH、[Puppet](http://linuxtoy.org/archives/puppet.html)、[Chef](http://www.opscode.com/chef/)、[SaltStack](http://linuxtoy.org/archives/saltstack.html)、[Ansible](http://www.ansible.com/)等。目前互联网企业使用最多的两款自动化配置工具Ansible和Saltstack。自动配置工具存在的初衷就是为了更方便、快捷的进行配置管理，它易于安装和使用、语法也非常简单易学。

本章向读者介绍Ansible工作原理、Ansible安装配置、生产环境模块讲解、Ansible企业场景案例、PlayBook剧本实战及Ansible性能调优等。

## 自动化运维工具简介

曾有媒体报道，Facebook一个运维人员管理上万台服务器，如果使用手工的方法去维护是很难做到的，基于自动化工具就可以轻松的实现管理上万台、甚至十万台。

如下为IT运维主流自动化管理工具Puppet、saltstack、Ansible各自优缺点：

### Puppet自动运维工具

Puppet是早期的Linux自动化运维工具，是一种Linux、Unix、Windows平台的集中配置管理系统，发展至今目前已经非常成熟，可以批量管理远程服务器，模块丰富，配置复杂，基于Ruby语言编写。最典型的C/S模式，需要安装服务端与客户端。

puppet采用C/S星状的结构，所有的客户端和一个或几个服务器交互，每个客户端周期的（默认半个小时）向服务器发送请求，获得其最新的配置信息，保证和该配置信息同步。

每个puppet客户端每半小时(可以设置)连接一次服务器端, 下载最新的配置文件,并且严格按照配置文件来配置客户端. 配置完成以后，puppet客户端可以反馈给服务器端一个消息，如果出错也会给服务器端反馈一个消息。

Puppet适用于服务器管理的整个过程，比如初始安装、配置、更新以及系统下线。

### Saltstack自动运维工具

Saltstack与Puppet均是C/S模式，需安装服务端与客户端，基于Python编写，加入MQ消息同步，可以使执行命令和执行结果高效返回，但其执行过程需等待客户端全部返回，如果客户端未及时返回或未响应的话，可能会导致部分机器没有执行结果。

### Ansible自动运维工具

Ansible与Saltstack均是基于Python语言开发，Ansible只需要在一台普通的服务器上运行即可，不需要在客户端服务器上安装客户端。因为Ansible是基于SSH远程管理，而Linux服务器大都离不开SSH，所以Ansible不需要为配置工作添加额外的支持。

Ansible安装使用非常简单，而且基于上千个插件和模块实现各种软件、平台、版本的管理，支持虚拟容器多层级的部署。很多读者在使用Ansible工具时，认为Ansible比Saltstatck执行效率慢，其实不是软件本身慢，是由于SSH服务慢，可以优化SSH连接速度及使用Ansible加速模块，满足企业上万台服务器的维护和管理。

## Ansible运维工具原理

Ansible是一款极为灵活的开源工具套件，能够大大简化Unix管理员的自动化配置管理与流程控制方式。它利用推送方式对客户系统加以配置，这样所有工作都可在主服务器端完成。其命令行机制同样非常强大，允许大家利用商业许可Web UI实现授权管理与配置。

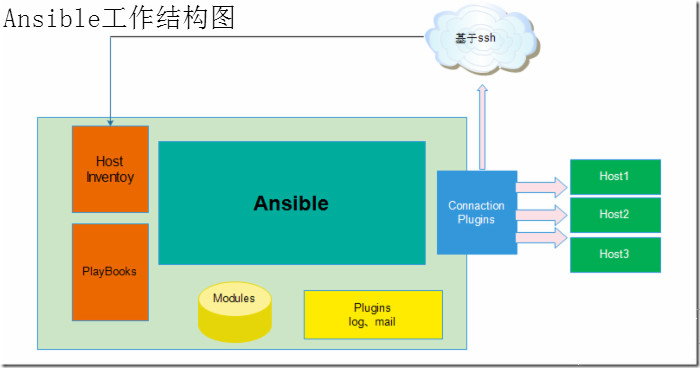
可以通过命令行或者GUI来使用Ansible，运行Ansible的服务器这里俗称“管理节点”;通过Ansible进行管理的服务器俗称“受控节点”。权威媒体报道Ansible于2015年被Red Hat公司1.5亿美元收购，新版Red Hat内置Ansible软件。

本书以Ansible为案例，基于Ansible构建企业自动化运维平台，实现大规模服务器的快速管理和部署。Ansible将平常复杂的配置工作变得简单，变得更加标准化更容易控制。

Ansible自动运维管理工具优点：

* 轻量级，更新时，只需要在操作机上进行一次更新即可;
* 采用SSH协议;
* 不需要去客户端安装agent;
* 批量任务执行可以写成脚本，而且不用分发到远程就可以执行;
* 使用python编写的，维护更简单;
* 支持sudo普通用户命令;
* 去中心化管理。

Ansible自动运维管理工具工作原理拓扑，如图21-1所示：

图21-1 Ansible工作原理图

## Ansible安装配置

Ansible 可以工作在Linux、BSD、Mac OS X 等平台，对Python环境的版本最低要求为Python2.6以上，如果操作系统Python软件版本为2.4，需要升级方可使用Ansible工具。

Red Hat、CentOS操作系统可以直接基于YUM工具自动安装Ansible，CentOS6.x或者CentOS7.x安装前，需先安装epel扩展源，代码如下：

|  |
| --- |
| rpm -Uvh http://mirrors.ustc.edu.cn/fedora/epel/6/x86\_64/epel-release-6-8.noarch.rpm  yum install epel-release -y  yum install ansible -y |

Ansible工具默认主目录为：/etc/ansible/，其中hosts文件为被管理机IP或者主机名列表，ansible.cfg为ansible主配置文件，roles为角色或者插件路径，默认该目录为空，如图21-2所示：

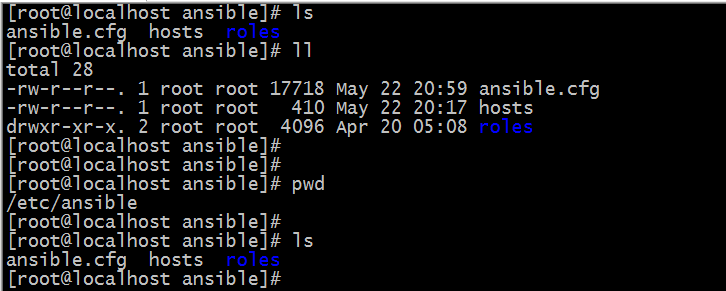


图21-2 Ansible主目录信息

Ansible远程批量管理，其中执行命令是通过Ad-Hoc来完成，也即点对点单条执行命令，能够快速执行，而且不需要保存执行的命令。默认hosts文件配置主机列表，可以配置分组，可以定义各种ip及规则，hosts列表默认配置如图21-3所示：

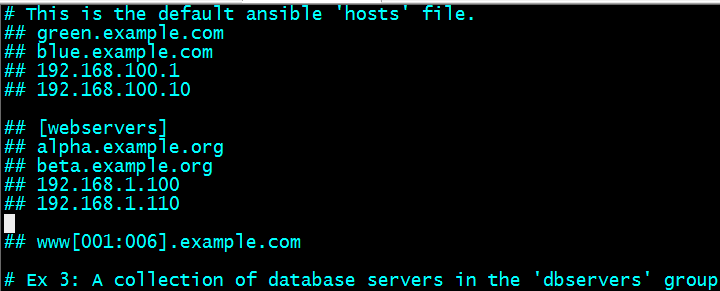


图21-3 Hosts主机列表文件内容

Ansible基于多模块管理，常用的Ansible工具管理模块包括：command、shell、script、yum、copy、File、async、docker、cron、mysql\_user、ping、sysctl、user、acl、add\_host、easy\_install、haproxy等。

可以使用ansible-doc -l|more查看ansible支持的模块，也可以查看每个模块的帮助文档，ansible-doc module\_name，如图21-4所示：

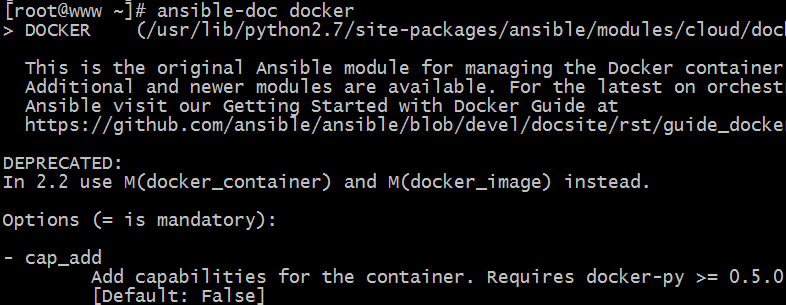


图21-4 Ansible-doc docker帮助信息

## Ansible工具参数详解

基于Ansible批量管理之前，需将被管理的服务器IP列表添加至/etc/ansible/hosts文件中，如图21-5添加4台被管理端IP地址，分成web和db两组，本机也可以是被管理机。

192.168.68.58 ansible\_user=jfedu ansible\_password=123456

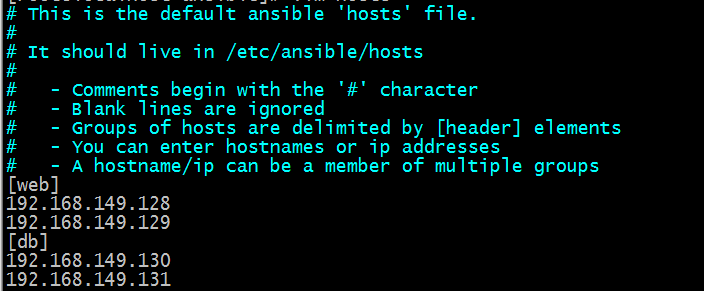


图21-5 Ansible Hosts主机列表

基于Ansible自动运维工具管理客户端案例操作，由于Ansible管理远程服务器基于SSH，在登录远程服务器执行命令时需要远程服务器的用户名和密码，也可以加入-k参数手动输入密码或者基于ssh-keygen生成免秘钥。

Ansible自动化批量管理工具主要参数如下：

|  |
| --- |
| -v,–verbose 打印详细模式;  -i PATH,–inventory=PATH 指定host文件路径;  -f NUM,–forks=NUM 指定fork开启同步进程的个数，默认5;  -m NAME,–module-name=NAME 指定module名称，默认模块command;  -a MODULE\_ARGS module模块的参数或者命令;  -k,–ask-pass 输入远程被管理端密码;  –sudo 基于sudo用户执行;  -K,–ask-sudo-pass 提示输入sudo密码与sudo一起使用;  -u USERNAME,–user=USERNAME 指定移动端的执行用户;  -C,–check 测试执行过程，不改变真实内容，相当于预演;  -T TIMEOUT, 执行命令超时时间，默认为10秒;  --version 查看Ansible软件版本信息。 |

## Ansible ping模块实战

Ansible最基础的模块为ping模块，主要用于判断远程客户端是否在线，用于ping本身服务器，返回值为changed、ping。

Ansible ping模块企业常用案例如下：

1. Ansible ping服务器状态，如图21-6所示：

|  |
| --- |
| ansible -k all -m ping |

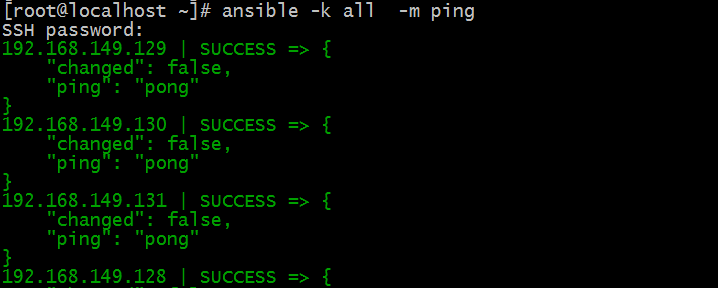
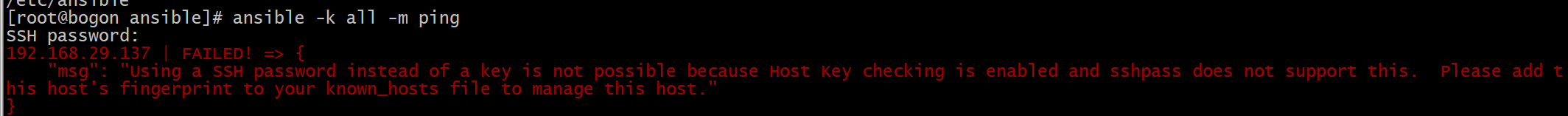


图21-6 Ansible ping服务器状态

备注：

1. vi /etc/ansible/ansible.cfg，将#host\_key\_checking = False改为host\_key\_checking = False（去掉注释），否则会报错  
   

## Ansible command模块实战

Ansible command模块为ansible默认模块，主要用于执行Linux基础命令，可以执行远程服务器命令执行、任务执行等操作。Command模块使用详解：

|  |
| --- |
| Chdir 执行命令前，切换到目录;  Creates 当该文件存在时，则不执行该步骤;  Executable 换用shell环境执行命令;  Free\_form 需要执行的脚本;  Removes 当该文件不存在时，则不执行该步骤;  Warn 如果在ansible.cfg中存在告警，如果设定了False，不会警告此行。 |

Ansible command模块企业常用案例如下：

1. Ansible command模块远程执行date命令，执行结果如图21-7所示：

|  |
| --- |
| ansible -k -i /etc/ansible/hosts all -m command -a "date" |

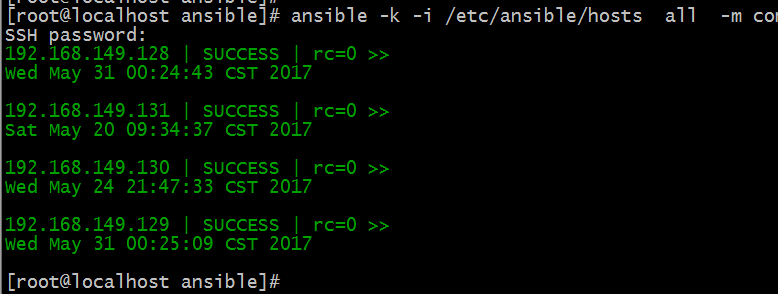


图21-7 Ansible command date命令执行结果

1. Ansible command模块远程执行ping命令，执行结果如图21-8所示：

|  |
| --- |
| ansible -k all -m command -a "ping -c 1 www.baidu.com" |

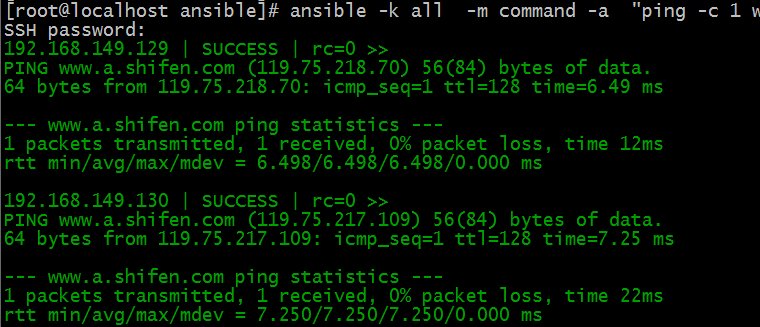
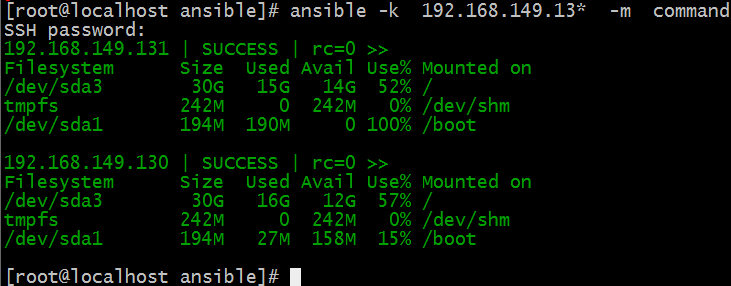


图21-8 Ansible command ping命令执行结果

1. Ansible Hosts正则模式远程执行df -h，执行结果如图21-9所示：

|  |
| --- |
| ansible -k 192.168.149.13\* -m command -a "df -h" |

图21-9 Ansible command df -h命令执行结果

## Ansible copy模块实战

Ansible copy模块主要用于文件或者目录拷贝，支持文件、目录、权限、用户组功能，copy模块使用详解：

|  |
| --- |
| src Ansible端源文件或者目录，空文件夹不拷贝;  content 用来替代src，用于将指定文件的内容，拷贝到远程文件内;  dest 客户端目标目录或者文件，需要绝对路径;  backup 拷贝之前，先备份远程节点上的原始文件;  directory\_mode 用于拷贝文件夹，新建的文件会被拷贝，而老旧的不会被拷贝;  follow 支持link文件拷贝;  force 覆盖远程主机不一致的内容;  group 设定远程主机文件夹的组名;  mode 指定远程主机文件及文件及的权限;  owner 设定远程主机文件夹的用户名。 |

Ansible copy模块企业常用案例如下：

1. Ansible copy模块操作，src表示源文件，dest表示目标目录或者文件，owner指定拥有者，执行结果如图21-10所示：

|  |
| --- |
| ansible -k all -m copy -a 'src=/etc/passwd dest=/tmp/ mode=755 owner=root' |

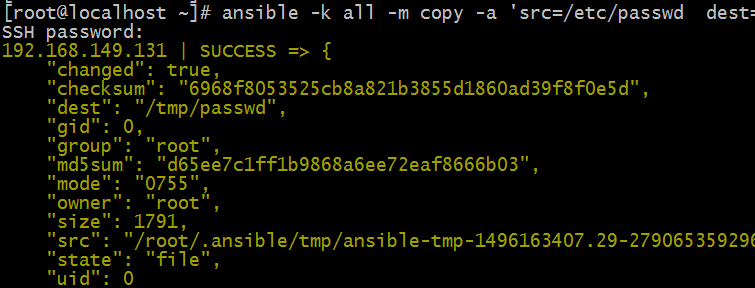


图21-10 Ansible copy拷贝文件

1. Ansible copy模块操作，content文件内容，dest目标文件，owner指定拥有者，执行结果如图21-11所示：

|  |
| --- |
| ansible -k all -m copy -a 'content="Hello World" dest=/tmp/jfedu.txt mode=755 owner=root' |

图21-11 Ansible copy追加内容

1. Ansible copy模块操作，content文件内容，dest目标文件，owner指定拥有者，backup=yes开启备份，执行结果如图21-12所示：

|  |
| --- |
| ansible -k all -m copy -a 'content="Hello World" dest=/tmp/jfedu.txt backup=yes mode=755 owner=root' |

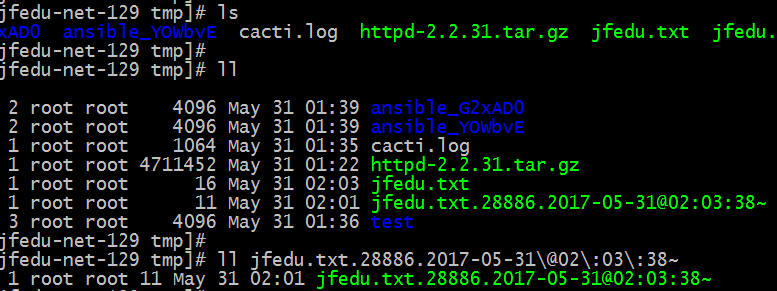


图21-12 Ansible copy客户端备份结果

## Ansible yum模块实战

Ansible yum模块主要用于软件的安装、升级、卸载，支持红帽.rpm软件的管理，YUM模块使用详解：

|  |
| --- |
| conf\_file 设定远程yum执行时所依赖的yum配置文件  disable\_gpg\_check 安装软件包之前是否坚持gpg key;  name 需要安装的软件名称，支持软件组安装;  update\_cache 安装软件前更新缓存;  enablerepo 指定repo源名称;  skip\_broken 跳过异常软件节点;  state 软件包状态，包括：installed、present、latest、absent、removed。 |

Ansible yum模块企业常用案例如下：

1. Ansible yum模块操作，name表示需安装的软件名称，state表示状态，常见state= installed表示安装软件，执行结果如图21-13所示：

|  |
| --- |
| ansible all -k -m yum -a "name=sysstat,screen state=installed" |

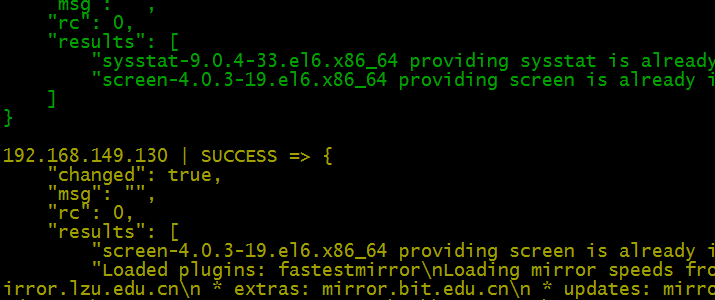


图21-13 Ansible YUM安装软件包

1. Ansible yum模块操作，name表示需安装的软件名称，state表示状态，常见state= installed表示安装软件，执行结果如图21-14所示：

|  |
| --- |
| ansible all -k -m yum -a "name=sysstat,screen state=absent" |

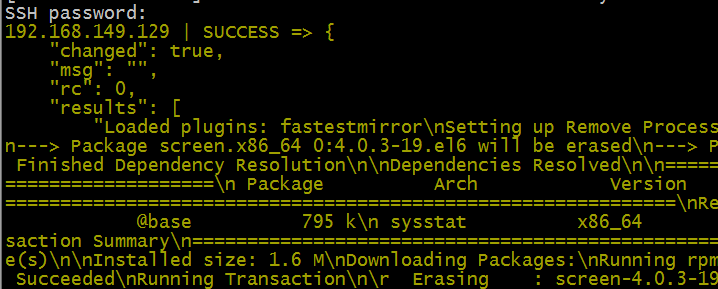
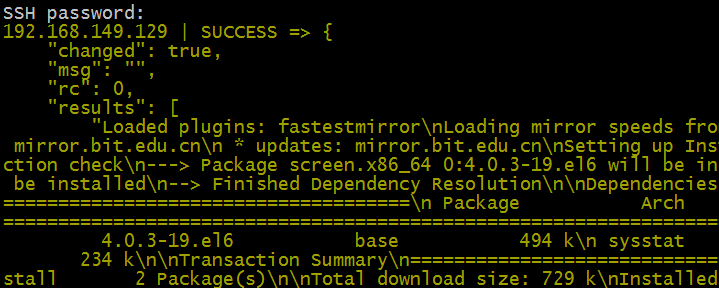


图21-14 Ansible YUM卸载软件包

1. Ansible yum模块操作，name表示需安装的软件名称，state表示状态，常见state= installed，表示安装软件，disable\_gpg\_check=no不检查key，执行结果如图21-15所示：

|  |
| --- |
| ansible 192.168.149.129 -k -m yum -a "name=sysstat,screen state=installed disable\_gpg\_check=no" |

图21-15 Ansible YUM安装软件包，不检查KEY

## Ansible file模块实战

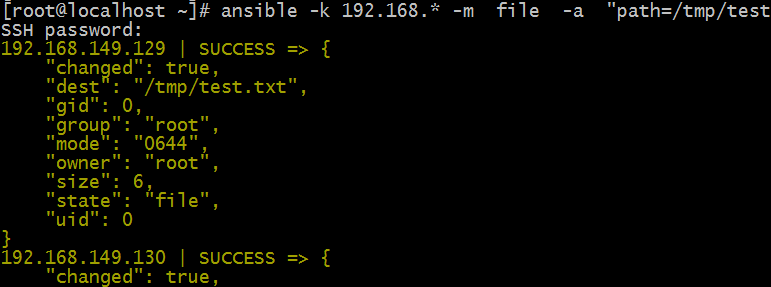
Ansible file模块主要用于对文件的创建、删除、修改、权限、属性的维护和管理，File模块使用详解：

|  |
| --- |
| src Ansible端源文件或者目录;  follow 支持link文件拷贝;  force 覆盖远程主机不一致的内容;  group 设定远程主机文件夹的组名;  mode 指定远程主机文件及文件及的权限;  owner 设定远程主机文件夹的用户名;  path 目标路径，也可以用dest,name代替;  state 状态包括：file、link、directory、hard、touch、absent;  attributes 文件或者目录特殊属性。 |

Ansible file模块企业常用案例如下：

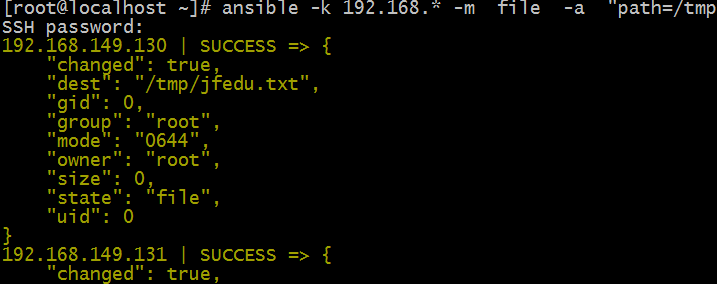
1. Ansible file模块操作，path表示目录的名称和路径， state=directory表示创建目录，执行结果如图21-16所示：

|  |
| --- |
| ansible -k 192.168.\* -m file -a "path=/tmp/`date +%F` state=directory mode=755" |

图21-16 Ansible file创建目录

1. Ansible file模块操作，path表示目录的名称和路径， state=touch表示创建文件，执行结果如图21-17所示：

|  |
| --- |
| ansible -k 192.168.\* -m file -a "path=/tmp/jfedu.txt state=touch mode=755" |

图21-17 Ansible file创建文件

## Ansible user模块实战

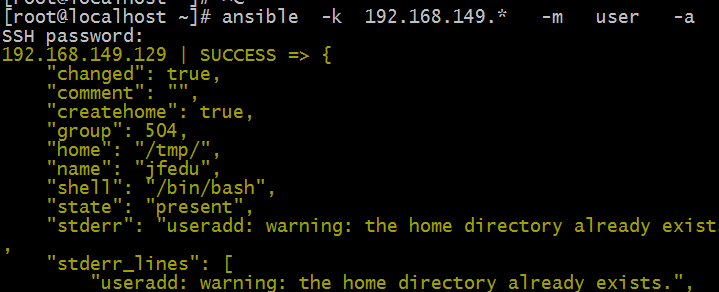
Ansible user模块主要用于操作系统用户、组、权限、密码等操作，user模块使用详解：

|  |
| --- |
| system 默认创建为普通用户，为yes则创建系统用户;  append 添加一个新的组;  comment 新增描述信息;  createhome 给用户创建家目录;  force 用于删除强制删除用户;  group 创建用户主组;  groups 将用户加入组或者附属组添加;  home 指定用户的家目录;  name 表示状态，是否create、remove、modify;  password 指定用户的密码，此处为加密密码;  remove 删除用户;  shell 设置用户的shell登录环境;  uid 设置用户id;  update\_password 修改用户密码;  state 用户状态，默认为present表示新建用户。 |

Ansible user模块企业常用案例如下：

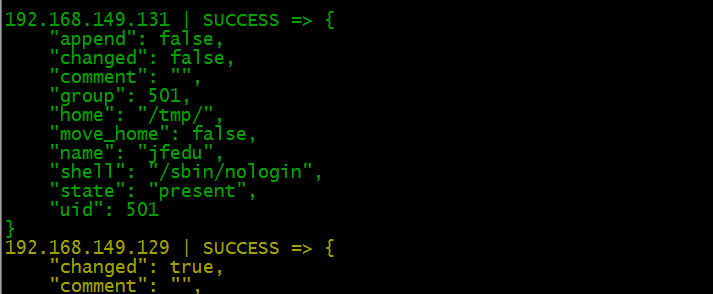
1. Ansible user模块操作，name表示用户名称，home表示其家目录，执行结果如图21-18所示：

|  |
| --- |
| ansible -k 192.168.149.\* -m user -a "name=jfedu home=/tmp/" |

图21-18 Ansible user创建新用户

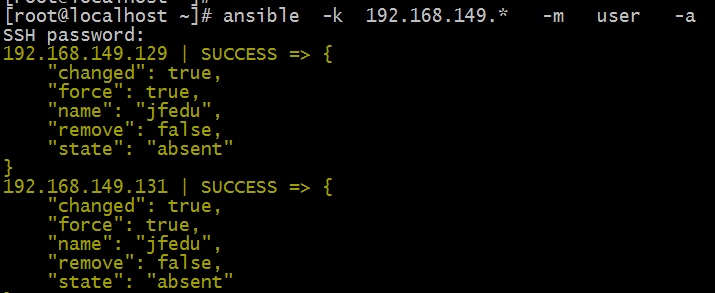
1. Ansible user模块操作，name表示用户名称，home表示其家目录，执行结果如图21-19所示：

|  |
| --- |
| ansible -k 192.168.149.\* -m user -a "name=jfedu home=/tmp/ shell=/sbin/nologin" |

图21-19 Ansible user指定Shell环境

1. Ansible user模块操作，name表示用户名称，state=absent表示删除用户，执行结果如图21-20所示：

|  |
| --- |
| ansible -k 192.168.149.\* -m user -a "name=jfedu state=absent force=yes" |

图21-20 Ansible user删除用户

生成密文密码

ansible all -i localhost, -m debug -a "msg={{ '密码信息(123456)' | password\_hash('sha512','加密效验信息(随便写)') }}"

-i localhost 在本地生成信息-m debug 调试模块

sha512 加密方式

# ansible all -i localhost, -m debug -a "msg={{ '123456' | password\_hash('sha512','sky01') }}"

localhost | SUCCESS => {

"msg": "$6$sky01$Ppe22caGxlbHwCR5biZ0oCcMa63McBIBgXD3RIkO080MckocOdUl2/SpKWUgPCdAOOE1Yjzyb5Oir2vgOjVwL/"}

再去修改创建密码(-a 后面请使用单引号'')

# ansible 10.0.0.11 -m user -a 'name=sky01 password=$6$sky01$Ppe22caGxlbHwCR5biZ0oCcMa63McBIBgXD3RIkO080MckocOdUl2/SpKWUgPCdAOOE1Yjzyb5Oir2vgOjVwL/'[WARNING]: The input password appears not to have been hashed. The 'password' argument must be encrypted for this module to work properly.

10.0.0.11 | CHANGED => {

"ansible\_facts": {

"discovered\_interpreter\_python": "/usr/bin/python"

},

"append": false,

"changed": true,

"comment": "",

"group": 1001,

"home": "/home/sky01",

"move\_home": false,

"name": "sky01",

"password": "NOT\_LOGGING\_PASSWORD",

"shell": "/bin/bash",

"state": "present",

"uid": 1001}

## Ansible cron模块实战

Ansible cron模块主要用于添加、删除、更新操作系统Crontab任务计划，Cron模块使用详解：

|  |
| --- |
| name 任务计划名称;  cron\_file 替换客户端该用户的任务计划的文件;  minute 分（ 0-59 ，\* ，\*/2 ）;  hour 时（ 0-23 ，\* ，\*/2 ）;  day 日（ 1-31 ，\* ，\*/2 ）;  month 月（ 1-12 ，\* ，\*/2 ）;  weekday 周（ 0-6 或 1-7 ，\* ）;  job 任何计划执行的命令，state要等于present;  backup 是否备份之前的任务计划;  user 新建任务计划的用户;  state 指定任务计划present、absent。 |

Ansible cron模块企业常用案例如下：

1. Ansible cron模块操作，基于cron模块，创建crontab任务计划，执行结果如图21-21所示：

|  |
| --- |
| ansible -k all -m cron -a "minute=0 hour=0 day=\* month=\* weekday=\* name='Ntpdate server for sync time' job='/usr/sbin/ntpdate 139.224.227.121'" |

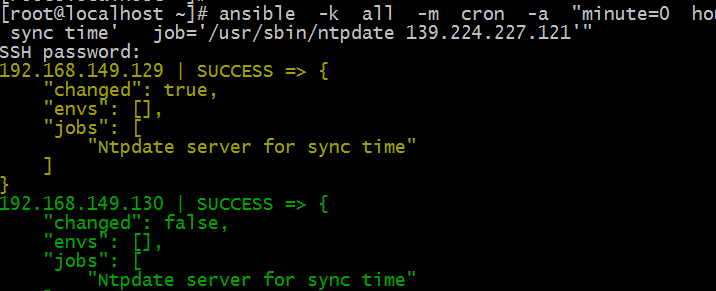
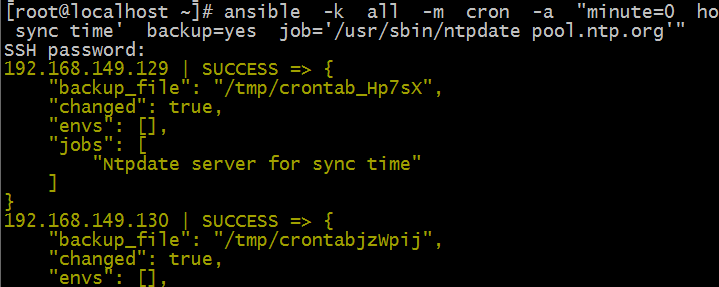


图21-21 Ansible cron添加任务计划

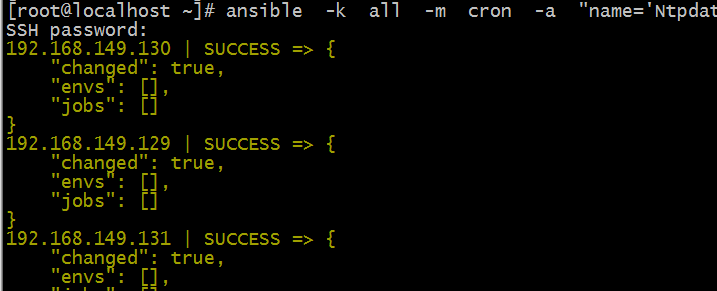
1. Ansible cron模块操作，基于cron模块，备份crontab任务计划，backup=yes表示开启备份，备份文件存放于客户端/tmp/，执行结果如图21-22所示：

|  |
| --- |
| ansible -k all -m cron -a "minute=0 hour=0 day=\* month=\* weekday=\* name='Ntpdate server for sync time' backup=yes job='/usr/sbin/ntpdate pool.ntp.org'" |

图21-22 Ansible cron删除任务计划

1. Ansible cron模块操作，基于cron模块，删除crontab任务计划，执行结果如图21-23所示：

|  |
| --- |
| ansible -k all -m cron -a "name='Ntpdate server for sync time' state=absent" |

图21-23 Ansible cron删除任务计划

## Ansible synchronize模块实战

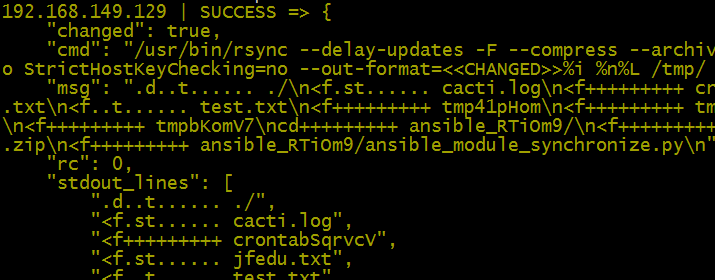
Ansible synchronize模块主要用于目录、文件同步，基于Rsync命令同步目录，Synchronize模块使用详解：

|  |
| --- |
| compress 开启压缩，默认为开启;  archive 是否采用归档模式同步，保证源和目标文件属性一致;  checksum 是否效验;  dirs 以非递归的方式传输目录;  links 同步链接文件;  recursive 是否递归yes/no;  rsync\_opts 使用rsync 的参数;  copy\_links 同步的时候是否复制连接;  delete 删除源中没有而目标存在的文件;  src 源目录及文件;  dest 目标目录及文件;  dest\_port 目标接受的端口;  rsync\_path 服务的路径，指定 rsync 命令来在远程服务器上运行;  rsync\_timeout 指定rsync操作的IP超时时间;  set\_remote\_user 设置远程用户名;  --exclude=.log 忽略同步.log结尾的文件;  mode 同步的模式，rsync同步的方式PUSH、PULL，默认都是推送push。 |

Ansible synchronize模块企业常用案例如下：

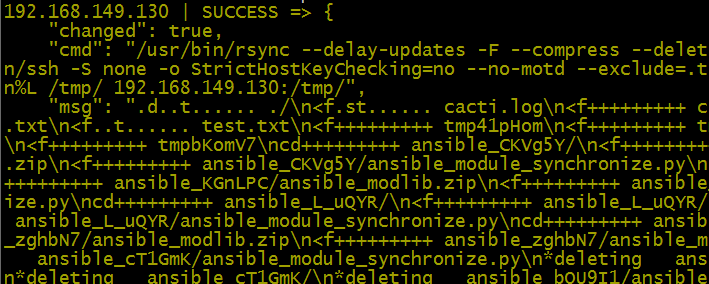
1. Ansible synchronize模块操作，src源目录、dest目标目录，执行结果如图21-24所示：

|  |
| --- |
| ansible -k all -m synchronize -a 'src=/tmp/ dest=/tmp/' |

图21-24 Ansible 目录同步

1. Ansible synchronize模块操作，src源目录、dest目标目录、compress=yes开启压缩、delete=yes数据一致、rsync\_opts同步参数、--exclude排除文件，执行结果如图21-25所示：

|  |
| --- |
| ansible -k all -m synchronize -a 'src=/tmp/ dest=/tmp/ compress=yes delete=yes rsync\_opts=--no-motd,--exclude=\*.txt' |

图21-25 Ansible 目录同步排除.txt文件

## Ansible shell模块实战

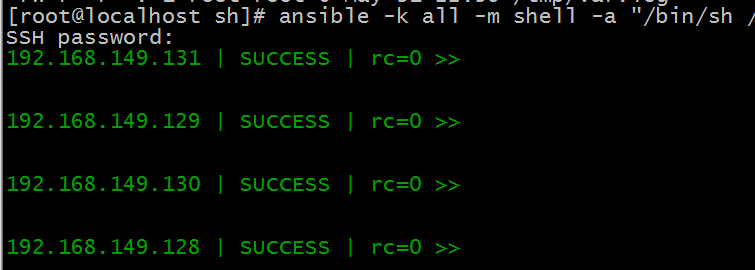
Ansible shell模块主要用于远程客户端上执行各种Shell命令或者运行脚本，远程执行命令通过/bin/sh环境来执行，支持比command更多的指令，Shell模块使用详解：

|  |
| --- |
| Chdir 执行命令前，切换到目录;  Creates 当该文件存在时，则不执行该步骤;  Executable 换用shell环境执行命令;  Free\_form 需要执行的脚本;  Removes 当该文件不存在时，则不执行该步骤;  Warn 如果在ansible.cfg中存在告警，如果设定了False，不会警告此行。 |

Ansible shell模块企业常用案例如下：

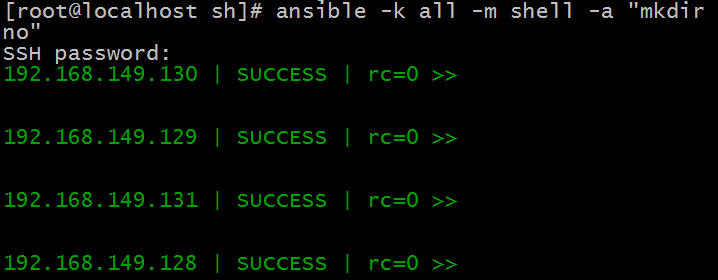
1. Ansible shell模块操作，-m shell指定模块为shell，远程执行Shell脚本，远程执行脚本也可采用script模块。并把执行结果追加至客户端服务器/tmp/var.log文件，执行结果如图21-26所示：

|  |
| --- |
| ansible -k all -m shell -a "/bin/sh /tmp/variables.sh >>/tmp/var.log" |

图21-26 Ansible shell远程执行脚本

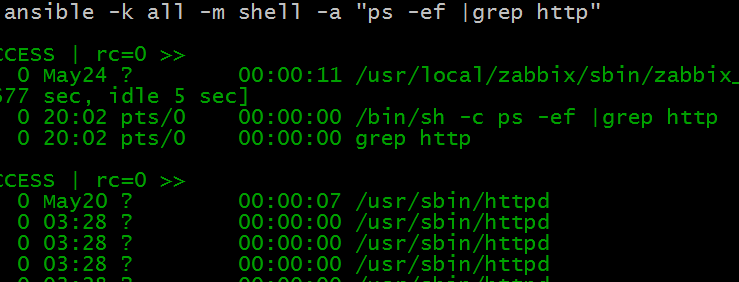
1. Ansible shell模块操作，远程执行创建目录命令，执行之前切换在/tmp目录，屏蔽警告信息，执行结果如图21-27所示：

|  |
| --- |
| ansible -k all -m shell -a "mkdir -p `date +%F` chdir=/tmp/ state=directory warn=no" |

图21-27 Ansible shell远程执行脚本

1. Ansible shell模块操作，-m shell指定模块为shell，远程客户端查看http进程是否启动，执行结果如图21-28所示：

|  |
| --- |
| ansible -k all -m shell -a "ps -ef |grep http" |

图21-28 Ansible shell远程查看进程

1. Ansible shell模块操作，-m shell指定模块为shell，远程客户端查看crontab任务计划，执行结果如图21-29所示：

|  |
| --- |
| ansible -k all -m shell -a "crontab -l" |

图21-29 Ansible shell远程查看任务计划

## Ansible service模块实战

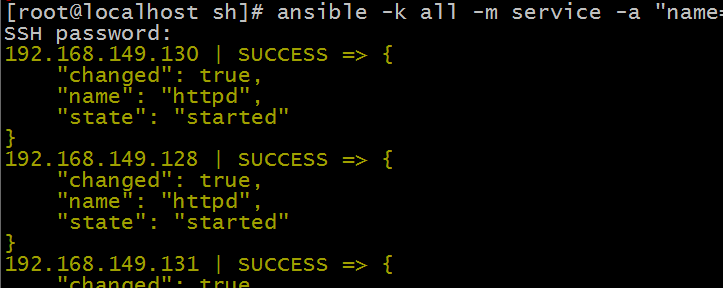
Ansible service模块主要用于远程客户端各种服务管理，包括启动、停止、重启、重新加载等，service模块使用详解：

|  |
| --- |
| enabled 是否开启启动服务;  name 服务名称;  runlevel 服务启动级别;  arguments 服务命令行参数传递;  state 服务操作状态，状态包括started, stopped, restarted, reloaded。 |

Ansible service模块企业常用案例如下：

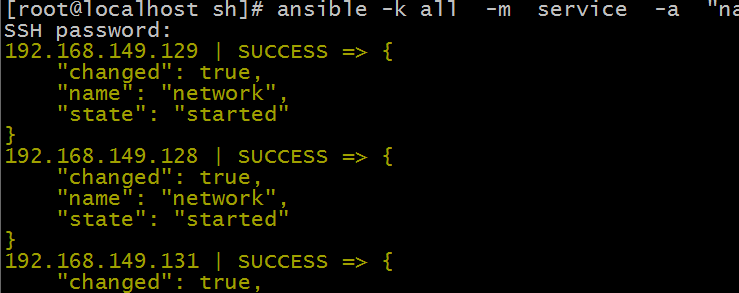
1. Ansible service模块操作，远程重启httpd服务，执行结果如图21-30所示：

|  |
| --- |
| ansible -k all -m service -a "name=httpd state=restarted" |

图21-30 Ansible service重启httpd服务

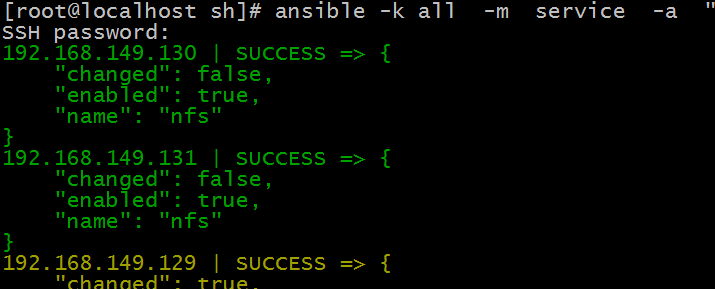
1. Ansible service模块操作，远程重启网卡服务，指定参数eth0，执行结果如图21-31所示：

|  |
| --- |
| ansible -k all -m service -a "name=network args=eth0 state=restarted" |

图21-31 Ansible service重启network服务

1. Ansible service模块操作，远程开机启动nfs服务，设置3,5级别自动启动，执行结果如图21-32所示：

|  |
| --- |
| ansible -k all -m service -a "name=nfs enabled=yes runlevel=3,5" |

图21-32 Ansible service开机启动nfs服务

## Ansible Playbook应用

如上使用Ad-hoc方式点对点命令执行，可以管理远程主机，如果服务器数量很多，配置信息比较多，还可以利用Ansible Playbook编写剧本、从而以非常简便的方式实现任务处理的自动化与流程化。

Playbook由一个或多个"play"组成的列表，play的主要功能Ansible中的Task定义好的角色，指定剧本对应的服务器组。

从根本上说，Task是一个任务，Task调用Ansible各种模块module，将多个paly组织在一个playbook剧本中，然后组成一个非常完整的流程控制集合。

基于Ansible Playbook还可以收集命令、可以创建任务集，这样能够大大降低管理工作的复杂程度，Playbook采用YAML语法结构，易于阅读、方便配置。

YAML（Yet Another Markup Language），是一种直观的能够被电脑识别的数据序列化格式，是一个可读性高并且容易被人类阅读，容易和脚本语言交互，用来表达资料序列的编程语言。它参考了其它多种语言，包括：XML、C语言、Python、Perl以及电子邮件格式RFC2822，是类似于标准通用标记语言的子集XML的数据描述语言，语法比XML简单很多。

YAML使用空白字符和分行来分隔资料，适合用 grep、Python、Perl、Ruby 操作。

1. YAML语言特性如下：

* 可读性强;
* 和脚本语言的交互性好;
* 使用实现语言的数据类型;
* 一致的信息模型;
* 易于实现;
* 可以基于流来处理;
* 可扩展性强。

1. Playbooks组件包括如下：

|  |
| --- |
| Target 定义playbook的远程主机组;  Variable 定义playbook使用的变量;  Task 定义远程主机上执行的任务列表;  Handler 定义task执行完成以后需要调用的任务，例如配置文件被改动，则启动handler任务重启相关联的服务。 |

1. Target常用参数如下：

|  |
| --- |
| hosts 定义远程主机组;  user 执行该任务的用户;  sudo 设置为yes的时候，执行任务的时候使用root权限;  sudo\_user 指定sudo普通用户;  connection 默认基于SSH连接客户端;  gather\_facks 获取远程主机facts基础信息。 |

1. Variable常用参数如下：

|  |
| --- |
| vars 定义格式，变量名:变量值;  vars\_files 指定变量文件;  vars\_prompt 用户交互模式自定义变量;  setup 模块去远程主机的值; |

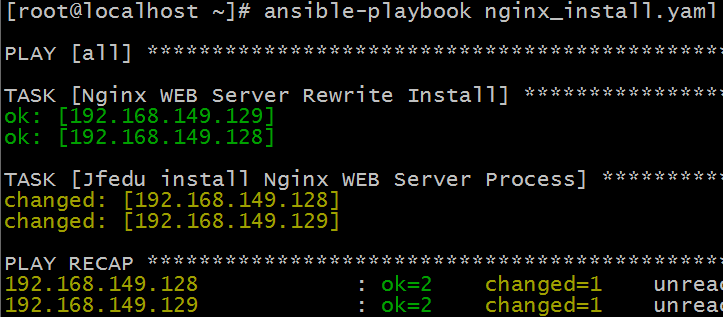
1. Task常用参数如下：

|  |
| --- |
| name 任务显示名称也即屏幕显示信息;  action 定义执行的动作;  copy 复制本地文件到远程主机;  template 复制本地文件到远程主机，可以引用本地变量;  service 定义服务的状态。 |

Ansible playbook案例演示如下：

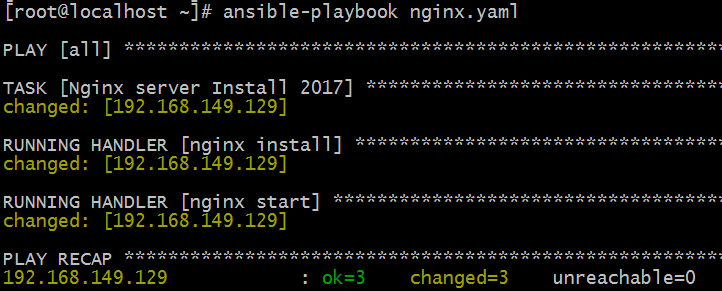
1. 远程主机安装Nginx WEB服务，playbook代码如下，执行结果如图21-33所示：

|  |
| --- |
| - hosts: all  remote\_user: root  tasks:  - name: Jfedu Pcre-devel and Zlib LIB Install.  yum: name=pcre-devel,pcre,zlib-devel state=installed  - name: Jfedu Nginx WEB Server Install Process.  shell: cd /tmp;rm -rf nginx-1.12.0.tar.gz;wget http://nginx.org/download/nginx-1.12.0.tar.gz;tar xzf nginx-1.12.0.tar.gz;cd nginx-1.12.0;./configure --prefix=/usr/local/nginx;make;make install |

图21-33 Ansible Playbook远程Nginx安装

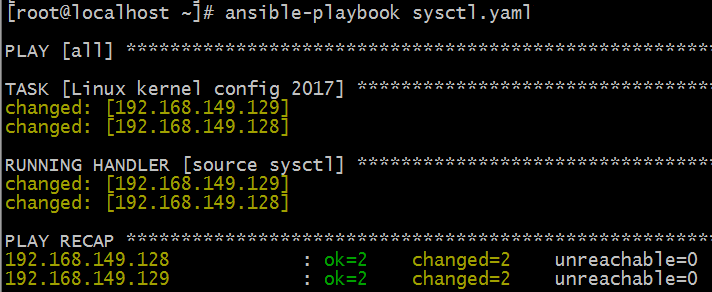
1. 检测远程主机Nginx目录是否存在，不存在则安装Nginx WEB服务，安装完并启动Nginx，playbook代码如下，执行结果如图21-34所示：

|  |
| --- |
| - hosts: all  remote\_user: root  tasks:  - name: Nginx server Install 2017  file: path=/usr/local/nginx/ state=directory  notify:  - nginx install  - nginx start  handlers:  - name: nginx install  shell: cd /tmp;rm -rf nginx-1.12.0.tar.gz;wget http://nginx.org/download/nginx-1.12.0.tar.gz;tar xzf nginx-1.12.0  .tar.gz;cd nginx-1.12.0;./configure --prefix=/usr/local/nginx;make;make install  - name: nginx start  shell: /usr/local/nginx/sbin/nginx |

图21-34 Ansible Playbook Nginx触发安装

1. 检测远程主机内核参数配置文件是否更新，如果更新则执行命令sysctl –p使内核参数生效，playbook代码如下，执行结果如图21-35所示：

|  |
| --- |
| - hosts: all  remote\_user: root  tasks:  - name: Linux kernel config 2017  copy: src=/data/sh/sysctl.conf dest=/etc/  notify:  - source sysctl  handlers:  - name: source sysctl  shell: sysctl -p |

图21-35 Ansible Playbook 内核参数优化

1. 基于列表items多个值创建用户，通过{{}}定义列表变量，with\_items选项传入变量的值，执行结果如图21-36（a）、21-36（b）所示：

|  |
| --- |
| - hosts: all  remote\_user: root  tasks:  - name: Linux system Add User list.  user: name={{ item }} state=present  with\_items:  - jfedu1  - jfedu2  - jfedu3  - jfedu4 |

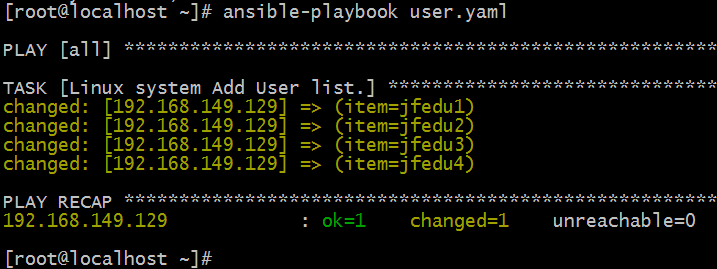


图21-36（a） Ansible Playbook item变量创建用户

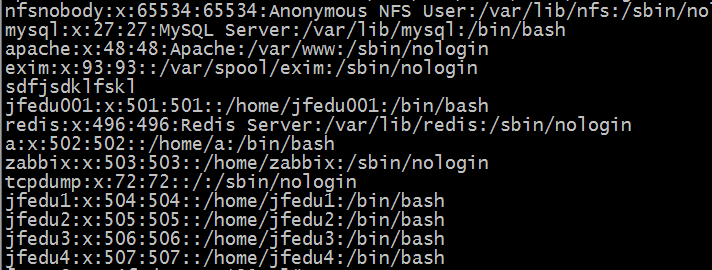


图21-36（b） Ansible Playbook item变量创建用户

1. Ansible Playbook可以自定义template模板文件，模板文件主要用于服务器需求不一致的情况，需要独立定义的，例如两台服务器安装了Nginx，安装完毕之后将服务器A的HTTP端口改成81，服务器B的HTTP端口改成82，基于tempalte模块轻松实现，方法步骤如下：
2. Ansible hosts文件指定不同服务器不同httpd\_port端口，代码如下：

|  |
| --- |
| [web]  192.168.149.128 httpd\_port=81  192.168.149.129 httpd\_port=82 |

1. Ansible 创建nginx.conf jinja2模板文件，cp nginx.conf nginx.conf.j2，并修改listen 80为listen {{httpd\_port}}，Nginx其他配置项不变，代码如下：

|  |
| --- |
| cp nginx.conf nginx.conf.j2  listen {{httpd\_port}}; |

1. Ansible playbook剧本yaml文件创建，代码如下：

|  |
| --- |
| - hosts: all  remote\_user: root  tasks:  - name: Nginx server Install 2017  file: path=/usr/local/nginx/ state=directory  notify:  - nginx install  - nginx config  handlers:  - name: nginx install  shell: cd /tmp;rm -rf nginx-1.12.0.tar.gz;wget http://nginx.org/download/nginx-1.12.0.tar.gz;tar xzf nginx-1.12.0  .tar.gz;cd nginx-1.12.0;./configure --prefix=/usr/local/nginx;make;make install  - name: nginx config  template: src=/data/sh/nginx.conf.j2 dest=/usr/local/nginx/conf/nginx.conf |

1. Ansible playbook执行剧本文件，如图21-37（a）、21-37（b）、21-37（c）所示：

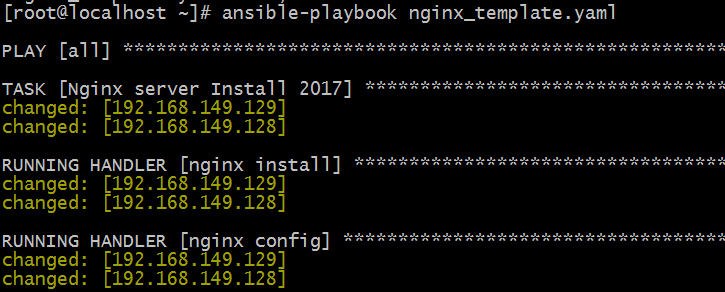


图21-37（a） Ansible Playbook 执行模板yaml

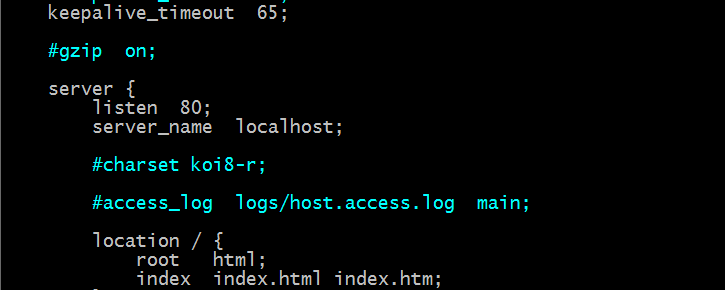


图21-37（b） 149.128服务器Nginx HTTP Port 80

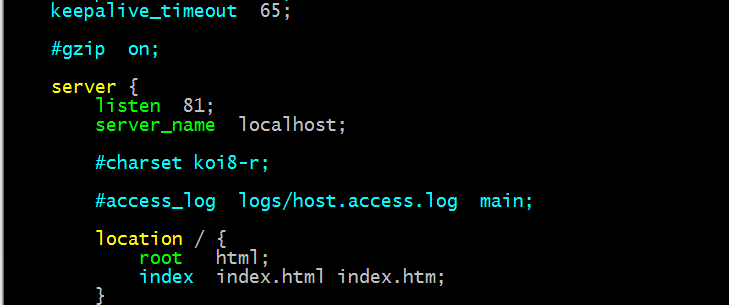


图21-37（c） 149.129服务器Nginx HTTP Port 81

## Ansible配置文件详解

Ansible默认配置文件为/etc/ansible/ansible.cfg，配置文件中可以对ansible进行各项参数的调整，包括并发线程、用户、模块路径、配置优化等，如下为Ansible.cfg常用参数详解：

|  |
| --- |
| [defaults] 通用默认配置段;  inventory = /etc/ansible/hosts 被控端IP或者DNS列表;  library = /usr/share/my\_modules/ Ansible默认搜寻模块的位置;  remote\_tmp = $HOME/.ansible/tmp Ansible远程执行临时文件;  forks = 5 并行进程数;  sudo\_user = root sudo远程执行用户名;  ask\_sudo\_pass = True 使用sudo，是否需要输入密码;  ask\_pass = True 是否需要输入密码;  remote\_port = 22 远程SSH端口;  module\_lang = C 模块和系统之间通信的语言;  host\_key\_checking = False 检查远程主机密钥;  #sudo\_exe = sudo sudo远程执行命令;  #sudo\_flags = -H 传递sudo之外的参数;  timeout = 10 SSH超时时间;  remote\_user = root 远程登陆用户名;  log\_path = /var/log/ansible.log 日志文件存放路径;  module\_name = command Ansible命令执行默认的模块;  #executable = /bin/sh 执行的Shell环境，用户Shell模块;  #system\_warnings = True 禁用系统运行ansible潜在问题警告;  #command\_warnings = False command模块Ansible默认发出警告;  #nocolor = 1 输出带上颜色区别，开启/关闭：0/1;  pipelining = False 开启pipe SSH通道优化; |

## Ansible性能调优

Ansible企业实战环境中，如果管理的服务器越来越多，Ansibe执行效率会变得比较慢，可以通过优化Ansible提供工作效率，由于Ansible基于SSH协议通信，SSH连接慢会导致整个基于Ansible执行变得缓慢，也需要对Openssh进行优化，具体优化的方法如下：

1. Ansible SSH 关闭秘钥检测

默认以SSH登录远程客户端服务器，会检查远程主机的公钥(public key)，并将该主机的公钥记录在~/.ssh/known\_hosts文件中。下次访问相同主机时，OpenSSH会核对公钥，如果公钥不同，OpenSSH会发出警告，如果公钥相同，则提示输入密码。

SSH对主机的public\_key的检查等级是根据StrictHostKeyChecking变量来设定的，StrictHostKeyChecking检查级别包括：no（不检查）、ask（询问）、yes（每次都检查）、False（关闭检查）。

Ansible配置文件中加入如下代码，即可关闭StrictHostKeyChecking检查：

|  |
| --- |
| host\_key\_checking = False |

1. OpenSSH连接优化

使用OpenSSH服务时，默认服务器端配置文件UseDNS=YES状态，该选项会导致服务器根据客户端的IP地址进行DNS PTR反向解析，得到客户端的主机名，然后根据获取到的主机名进行DNS正向A记录查询，并验证该IP是否与原始IP一致。关闭DNS解析代码如下：

|  |
| --- |
| sed -i '/^GSSAPI/s/yes/no/g;/UseDNS/d;/Protocol/aUseDNS no' /etc/ssh/sshd\_config  /etc/init.d/sshd restart |

1. SSH pipelining加速Ansible

SSH pipelining是一个加速 Ansible 执行速度的简单方法，SSH pipelining 默认是关闭的，关闭是为了兼容不同的 sudo 配置，主要是requiretty 选项。

如果不使用Sudo建议开启该选项，打开此选项可以减少Ansible 执行没有文件传输时，SSH在被控机器上执行任务的连接数。使用Sudo操作的时候, 必须在所有被管理的主机上将配置文件/etc/sudoers中requiretty选项禁用。

|  |
| --- |
| sed -i '/^pipelining/s/False/True/g' /etc/ansible/ansible.cfg |

1. Ansible Facts缓存优化

Ansible-playbook 在执行过程中，默认会执行Gather facts，如果不需要获取客户端的fact数据的话，可以关闭获取fact数据功能，关闭之后可以加快ansible-playbook的执行效率。如需关闭fact功能，在playbook yaml文件中加入如下代码即可：

|  |
| --- |
| gather\_facts: no |

Ansible facts组件主要用于收集客户端设备的基础静态信息，这些信息可以在做配置管理的时候方便引用。Facts信息直接当做Ansible Playbook变量信息进行引用，通过定制facts以便收集我们想要的信息，同时可以通过Facter和Ohai来拓展facts信息，也可以将facts信息存入Redis缓存中，如下为Facts使用Redis缓存的步骤。

1. 部署Redis服务

|  |
| --- |
| wget http://download.redis.io/releases/redis-2.8.13.tar.gz  tar zxf redis-2.8.13.tar.gz  cd redis-2.8.13  make PREFIX=/usr/local/redis install  cp redis.conf /usr/local/redis/ |

将/usr/local/redis/bin/目录加入至环境变量配置文件/etc/profile末尾，然后Shell终端执行source /etc/profile让环境变量生效。

|  |
| --- |
| export PATH=/usr/local/redis/bin:$PATH |

启动及停止Redis服务命令：

|  |
| --- |
| nohup /usr/local/redis/bin/redis-server /usr/local/redis/redis.conf & |

1. 安装Python Redis模块

|  |
| --- |
| easy\_install pip  pip install redis |

1. Ansible整合Redis配置

在配置文件/etc/ansible/ansible.cfg中defaluts段中加入代码，如果redis密码为admin，则开启admin密码行：

|  |
| --- |
| gathering = smart  fact\_caching = redis  fact\_caching\_timeout = 86400  fact\_caching\_connection = localhost:6379  #fact\_caching\_connection = localhost:6379:0:admin |

1. 测试Redis缓存

Ansible-playbook执行nginx\_wget.yaml剧本文件，如图21-38所示：

|  |
| --- |
| ansible-playbook nginx\_wget.yaml |

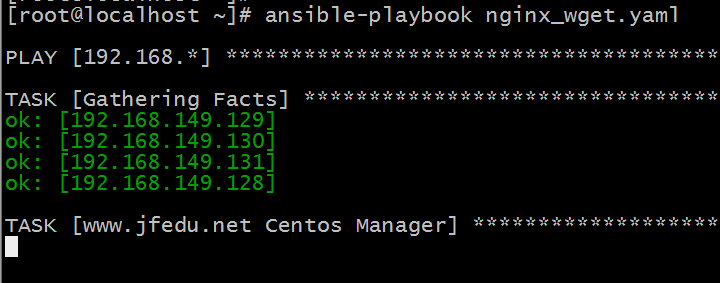


图21-38 ansible playbook执行yaml

检查Redis服务器，facts key已存入Redis中，如图21-39所示：

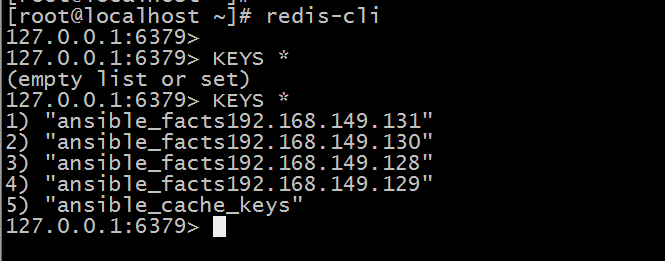


图21-39 Redis缓存服务器缓存facts主机信息

1. ControlPersist SSH优化

ControlPersist 特性需要高版本的SSH支持，CentOS6默认是不支持的，如果需要使用，需要升级Openssh。

ControlPersist 即持久化的Socket，一次验证多次通信。并且只需要修改SSH客户端配置，也即Ansible被管理主机。

可使用YUM或者源码编译升级OpenSSH服务，升级完毕ControlPersist的设置办法如下，在其用户的家目录创建config文件，如果ansible以root用户登录客户端，至需要在客户端的/root/.ssh/config文件中添加如下代码即可：

|  |
| --- |
| Host \*  Compression yes  ServerAliveInterval 60  ServerAliveCountMax 5  ControlMaster auto  ControlPath ~/.ssh/sockets/%r@%h-%p  ControlPersist 4h |

开启ControlPersist 特性后，SSH 在建立sockets后，节省了每次验证和创建的时间，对Ansible执行速度提升是非常明显的。