

Ansible简单介绍



jie0112 (/u/6800627be4d5) + 关注

2017.09.23 09:04* 字数 1114 阅读 340 评论 0 喜欢 0

(/u/6800627be4d5)

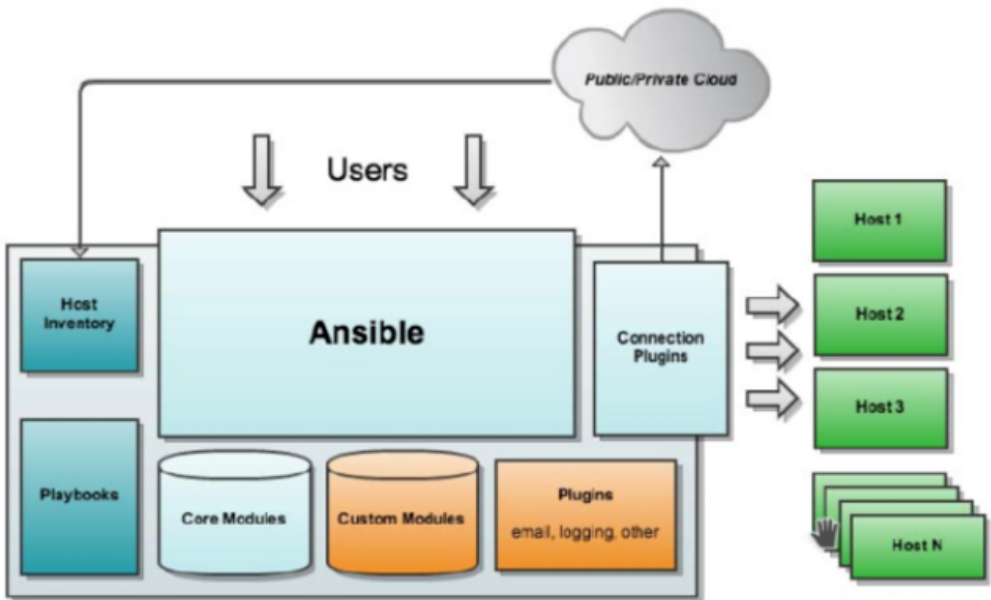
Ansible

ansible是个什么东西呢？官方的title是“Ansible is Simple IT Automation”——简单的自动化IT工具。这个工具的目标有这么几项：自动化部署APP；自动化管理配置项；自动化的持续交互；自动化的（AWS）云服务管理。所有的这几个目标从本质上来说都是在一个台或者几台服务器上，执行一系列的命令而已。通俗的说就是批量的在远程服务器上执行命令。当然，最主要的是它是基于 paramiko 开发的。这个paramiko是什么呢？它是一个纯Python实现的ssh协议库。因此fabric和ansible还有一个共同点就是不需要在远程主机上安装client/agents，因为它们是基于ssh来和远程主机通讯的。简单归纳一下：Ansible:—基于 Python paramiko 开发，分布式，无需客户端，轻量级，配置语法使用YMAL 及 Jinja2模板语言，更强的远程命令执行操作
类似的自动化运维工具有很多常用的还有：

Puppet

—基于 Ruby 开发，采用 C/S 架构，扩展性强，基于 SSL，远程命令执行相对较弱

Ansible工作机制



由上面的图可以看到 Ansible 的组成由 5 个部分组成：

Ansible： 核心

Modules： 包括 Ansible 自带的核心模块及自定义模块

Plugins： 完成模块功能的补充，包括连接插件、邮件插件等

Playbooks： 剧本(编排好一步一步的执行)；定义 Ansible 多任务配置文件，有 Ansible 自动执行



Inventory：定义 Ansible 管理主机的清单

一、ansible的安装

一是从官网下载最新的版本编译安装，一是直接从配好的yum 仓库安装。如果不追求新功能，要求稳定，一般我们直接从yum仓库安装即可。

这里的实验环境为centos7.3主机

(/apps/redi
utm_sourc
banner-clic

```
[root@centos7 ~]#yum info ansible #这里显示的版本为3.6.1
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
Installed Packages
Name           : ansible
Arch            : noarch
Version         : 2.3.1.0
Release        : 1.el7
Size           : 27 M
.....
```

安装部署ansible

ansible是基于ssh协议的首先让ansible主机能通过ssh密钥连接各主机

1.安装

```
[root@centos7 ~]#yum -y install ansible
1.创建ssh密钥连接
[root@centos7 ~]# ssh-keygen -t rsa -P "" #生成私钥
2.传递给后端目标主机
[root@centos7 ~]#ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.18.97
[root@centos7 ~]#ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.18.98
[root@centos7 ~]#vim /etc/ansible/hosts #编辑添加ansible的hosts配置文件
[webservs] #添加两个后端web服务主机
192.168.18.97
192.168.18.98
```

2.探测是否能通

```
[root@centos7 ansible]#ansible all -m ping #这里的all是定义的所有主机，也可以写自己在h
192.168.18.98 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
192.168.18.97 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

Ansible命令参数介绍

```
[root@centos7 ansible]#ansible --help #这里常用命令的介绍
-v,--verbose # 详细模式，如果命令执行成功，输出详细的结果(-vv -vvv -vvvv)
-i PATH,--inventory=PATH #指定host文件的路径，默认是/etc/ansible/hosts
-f NUM,--forks=NUM # NUM是指定一个整数，默认是5，指定fork开启同步进程的个数。
-m NAME,--module-name=NAME #指定使用的module名称，默认是command
-m DIRECTORY,--module-path=DIRECTORY #指定module的目录来加载module，默认是/usr/share/ar
-a,MODULE_ARGS #指定module模块的参数
-k,--ask-pass # 提示输入ssh的密码，而不是使用基于ssh的密钥认证
-sudo # 指定使用sudo获得root权限
-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 #测试此命令执行会改变什么内容，不会真正的去执行
```



ansible几个常用模块介绍及使用

```
[root@centos7 ansible]#ansible-doc -l #列出模块
```

(/apps/redi
utm_sourc
banner-clc

1.group : 组

```
[root@centos7 ansible]#ansible-doc -s group 查看组
- name: Add or remove groups
  action: group
    gid                # 组gid
    name=              # Name 组名.
    state              # present : 创建 absent : 删除
    system             #默认系统组no,
```

创建组:

```
[root@centos7 ansible]#ansible all -m group -a "gid=3000 name=mygrp state=present sy
192.168.18.98 | SUCCESS => {
  "changed": true,
  "gid": 3000,
  "name": "mygrp",
  "state": "present",
  "system": false
}
192.168.18.97 | SUCCESS => {
  "changed": true,
  "gid": 3000,
  "name": "mygrp",
  "state": "present",
  "system": false
}
192.168.18.97 ,192.168.18.98主机上查看
[root@centos7 ~]#tail -1 /etc/group
mygrp:x:3000: #都创建成功
```

删除组:

```
[root@centos7 ansible]#ansible all -m group -a "gid=3000 name=mygrp state=absent" #
```

2.user:用户

```
[root@centos7 ansible]#ansible-doc -s user #用户
name=      #名字
comment:   #注释信息
expires    #过期时间
group      #基本组
groups     #附加组
home       #家目录路径
move_home  #原来家目录及文件是否移动过来
passwd     #定义密码
shell      #选择shell
state      # 创建
system     #系统用户
uid        #uid
```

创建用户



```
[root@centos7 ansible]#ansible all -m user -a"uid=5000 name=testuser state=present c
192.168.18.98 | SUCCESS => {
    "changed": true,
    "comment": "",
    "createhome": true,
    "group": 5000,
    "groups": "mygrp",
    "home": "/home/testuser",
    "name": "testuser",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 5000
}
192.168.18.97 | SUCCESS => {
    . . . .
}
```

(/apps/redi
utm_sourc
banner-clic

删除

```
ansible all -m user -a "name=testuser state=absent"
```

3.copy:

```
[root@centos7 ansible]#ansible-doc -s copy #拷贝
dest=    #目标路径
src      #源 路径 是目录: 做递归 如果带/ 复制目录里面内容, 不带/ 复制目录和文件
mode     #权限
content  #生成文件
```

简单复制文件

```
[root@centos7 ~]#ansible all -m copy -a "src=/etc/fstab dest=/tmp/fstab.ansible mode=
[root@centos7 ~]#ansible all -m copy -a "src=/etc/pam.d/ dest=/tmp/" #只复制里面内容
[root@centos7 ~]#ansible all -m copy -a "src=/etc/pam.d dest=/tmp/" #复制目录及内容
[root@centos7 ~]#ansible all -m copy -a "content='hi there' dest=/tmp/ansib.txt" #生
[root@centos7 ~]#ansible all -m copy -a "content='hi there' dest=/tmp/ansib.txt owne
```

4.command :

```
[root@centos7 ~]#ansible-doc -s command #执行命令
chdir    #切换目录执行
executable #不适应默认shell使用此项
free_form= #自由格式执行
[root@centos7 ~]#ansible all -m command -a "ifconfig" #使用命令,
[root@centos7 ~]#ansible all -m command -a "chdir=/var/tmp mkdir hi.dir" #创建一个目
[root@centos7 ~]#ansible all -m command -a "echo jie |passwd --stdin testuser" #cc
192.168.18.97 | SUCCESS | rc=0 >>
    jie |passwd --stdin testuser
[root@centos7 ~]#ansible all -m shell -a "echo jie |passwd --stdin testuser" #做命令
192.168.18.98 | SUCCESS | rc=0 >>
Changing password for user testuser.
passwd: all authentication tokens updated successfully.
```

5.file:



```
[root@centos7 ~]#ansible-doc -s file #文件
path= #目标路径
start #创建文件 directory目录 file 文件
src #符号链接
mode #属主属组
[root@centos7 ~]#ansible all -m file -a "path=/var/tmp/hello.dir state=directory" #/
[root@centos7 ~]#ansible all -m file -a "src=/var/tmp/fstab.ansible path=/var/tmp/fs
```

(/apps/redi
utm_sourc
banner-clc

6.cron:

```
[root@centos7 ~]#ansible-doc -s cron #创建任务计划
day #天
hour #小时
job #任务
name #任务名称
state #present 添加 absent 删除
[root@centos7 ~]#ansible all -m cron -a "minute=*/3 job='usr/sbin/update 172.16.0.1
[root@centos7 tmp]#crontab -l
#Ansible: renwu
*/3 * * * * usr/sbin/update 172.16.0.1 &> /dev/nul
[root@centos7 ~]#ansible all -m cron -a "minute=*/3 job='usr/sbin/update 172.16.0.1
```

7.yum :

```
[root@centos7 ~]#ansible-doc -s yum #安装程序包
state #安装 ( present或installed或latest) 卸载 (absent或removed)
disablerepo #禁用某个仓库
enablerepo #启用某个仓库
disable_gpg_check #禁用gpg检测
[root@centos7 ~]#ansible all -m yum -a "name=nginx state=installed" #安装程序
```

8.service:

```
[root@centos7 ~]#ansible-doc -s service #服务
enabled #开机启用
name= #服务名称
pattern #过滤字符
runlevel #哪个级别启用
state # started 启动 stopped停止 restarted 重启 reloaded重载
[root@centos7 ~]#ansible all -m service -a "name=nginx state=started" #启动服务
[root@centos7 ~]#ansible all -m service -a "name=nginx state=stopped" #服务停止
```

9.script

```
[root@centos7 ~]#ansible-doc -s script #脚本
[root@centos7 ~]#vim /tmp/test.sh
#!/bin/bash
echo "test script" > /tmp/1.txt
[root@centos7 ~]#ansible all -m script -a "/tmp/test.sh" #执行脚本
```

Playbook:

Playbooks 是一个不同于使用Ansible命令行执行方式的模式，其功能更强大灵活。简单来说，playbook是一个非常简单的配置管理和多主机部署系统，不同于任何已经存在的模式，可作为一个适合部署复杂应用程序的基础。Playbook可以定制配置，可以按照指定的操作步骤有序执行，支持同步和异步方式。值得注意的是playbook是通过YAML格式来进行描述定义的。

yaml (<https://link.jianshu.com?t=https://zh.wikipedia.org/wiki/YAML>)用法：可参考维基百



科 或官网文档<http://docs.ansible.com/ansible/latest/YAMLSyntax.html>
(<https://link.jianshu.com?t=http://docs.ansible.com/ansible/latest/YAMLSyntax.html>)

Playbook的核心元素：

Hosts: 主机
Tasks: 任务列表
Variables: 变量
Templates: 包含了模板语法的文本文件;
Handlers: 由特定条件触发的任务;

(/apps/redi
utm_sourc
banner-clic

1.创建playbook

```
[root@centos7 ansible]#mkdir playbooks #创建剧本
[root@centos7 ansible]#cd playbooks
[root@centos7 playbooks]#vim 1.yaml
- host: 192.168.18.97 #主机
#也可以写成- hosts: all 或 - hosts: webservs
remote_user: root #使用的用户
tasks: #任务
- name: install redis #任务名称1
  yum: name=redis state=latest
- name: start redis #任务名称2
  service: name=redis state=started
[root@centos7 playbooks]#ansible-playbook --syntax-check 1.yaml #语法检查
[root@centos7 playbooks]#ansible-playbook -C 1.yaml #测试安装
#paly 让哪台主机执行
PLAY [all] *****
#收集信息在ansible
TASK [setup] *****
#安装Redis
TASK [install redis] *****
#启动Redis
TASK [start redis] *****
to retry, use: --limit @/root/playbooks/1.retry
#结果
PLAY RECAP *****
注意: 按每项任务执行, 不是每个主机执行完才执行另外主机!
```

2.对配置文件的修改

如不使用Redis的默认配置文件, 自己修改。可从别机拷一份配置文件过来做修改, 然后利用ansible部署。

从单台主机拷贝贝源文件到ansible主机

```
ansible 192.168.18.97 -m fetch -a "src=/etc/redis.conf dest=."/
```

编辑playbook里的1.yaml,假设redis.conf已修改好(修改的是bind 端口)。



```
[root@centos7 playbooks]#vim 1.yaml
- hosts: all
  remote_user: root
  tasks:
    - name: install redis
      yum: name=redis state=latest
    - name: copy config file
      copy: src=/root/playbooks/redis.conf dest=/etc/redis.conf owner=redis
    - name: start redis
      service: name=redis state=started
[root@centos7 playbooks]#ansible-playbook --syntax-check 1.yaml #检查语法
[root@centos7 playbooks]#ansible-playbook -C 1.yaml #测试安装
[root@centos7 playbooks]#ansible-playbook 1.yaml #安装
此时后端主机配置文件改变，但服务状态重启，还是原来的状态，端口没有变化
```

3.此时用到handlers:由特定条件触发的任务；

handlers:

任务，在特定条件下触发；

接收到其它任务的通知时被触发；

notify: HANDLER TASK NAME

```
[root@centos7 playbooks]#cp 1.yaml 2.yaml
[root@centos7 playbooks]#vim 2.yaml
- hosts: all
  remote_user: root
  tasks:
    - name: install redis
      yum: name=redis state=latest
    - name: copy config file
      copy: src=/root/playbooks/redis.conf dest=/etc/redis.conf owner=redis
      notify: restart redis
    - name: start redis
      service: name=redis state=started
  handlers: #接收到其它任务的通知时被触发；
    - name: restart redis
      service: name=redis stat=restarted
[root@centos7 playbooks]#ansible-playbook --syntax-check 2.yaml
[root@centos7 playbooks]#ansible-playbook -C 2.yaml
[root@centos7 playbooks]#ansible-playbook 2.yaml
改变的配置文件，触发重启服务
```

4..tags: 标签

做标签:只对有标签项作出操作,则可继续修改



```
[root@centos7 playbooks]#vim 2.yaml
- hosts: all
  remote_user: root
  tasks:
    - name: install redis
      yum: name=redis state=latest
    - name: copy config file
      copy: src=/root/playbooks/redis.conf dest=/etc/redis.conf owner=redis
      notify: restart redis
      tags: configfile
    - name: start redis
      service: name=redis state=started
  handlers:
    - name: restart redis
      service: name=redis state=restarted
[root@centos7 playbooks]#ansible-playbook --syntax-check 2.yaml
[root@centos7 playbooks]#ansible-playbook -C 2.yaml
[root@centos7 playbooks]#ansible-playbook -t configfile 2.yaml #对标签操作
```

(/apps/redi
utm_sourc
banner-cli

5.variables

- (1) facts: 可直接调用;
注意: 可使用setup模块直接获取目标主机的facters;
- (2) 用户自定义变量:
 - (a) ansible-playbook命令的命令行中的
-e VARS, --extra-vars=VARS
 - (b) 在playbook中定义变量的方法:


```
vars:
  - var1: value1
  - var2: value2
```

 变量引用: {{ variable }}
- (3) 通过roles传递变量;
- (4) Host Inventory
 - (a) 用户自定义变量
 - (i) 向不同的主机传递不同的变量;


```
IP/HOSTNAME variable=value var2=value2
```
 - (ii) 向组中的主机传递相同的变量;


```
[groupname:vars]
variable=value
```
 - (b) inventory参数
 用于定义ansible远程连接目标主机时使用的参数, 而非传递给playbook的变量;


```
ansible_ssh_host
ansible_ssh_port
ansible_ssh_user
ansible_ssh_pass
ansbile_sudo_pass
...
```

6.templates:模板



文本文件，嵌套有脚本（使用模板编程语言编写）

Jinja2:

字面量:

字符串: 使用单引号或双引号;

数字: 整数, 浮点数;

列表: [item1, item2, ...]

元组: (item1, item2, ...)

字典: {key1:value1, key2:value2, ...}

布尔型: true/false

算术运算:

+, -, *, /, //, %, **

比较操作:

==, !=, >, >=, <, <=

逻辑运算:

and, or, not

(/apps/redi
utm_sourc
banner-clic

列1.

```
模板配置文件 :
nginx.conf.j2
    worker_processes {{ ansible_processor_vcpus }};
    listen {{ http_port }};

- hosts: webservs
  remote_user: root
  tasks:
    - name: install nginx
      yum: name=nginx state=present
    - name: install conf file
      template: src=files/nginx.conf.j2 dest=/etc/nginx/nginx.conf
      notify: restart nginx
      tags: instconf
    - name: start nginx service
      service: name=nginx state=started
  handlers:
    - name: restart nginx
      service: name=nginx state=restarted
```

例2

```
模板配置文件 :
cp /root/playbook/redis.conf{,.j2}
vim /etc/playbook/redis.conf.j2
bind {{ ansible_ens33.ipv4.address }} //调用网卡

- hosts: all
  remote_user: root
  tasks:
    - name: install config file
      template: src=/root/playbooks/redis.conf.j2 dest=/tmp/redis.conf
```

7.条件测试和循环（迭代，需要重复执行的任务）



(/apps/redi
utm_sourc
banner-clic

```
[root@centos7 playbooks]#vim os.yaml
- hosts: webservs
  remote_user: root
  tasks:
    - name: install httpd
      yum: name=httpd state=latest
      when: ansible_os_family == "RedHat"
    - name: install apache2
      apt: name=apache2 state=latest
      when: ansible_os_family == "Debian"
[root@centos7 playbooks]#ansible-playbook --syntax-check os.yaml
迭代安装:
[root@centos7 playbooks]#vim 8.yaml
- hosts: webservs
  remote_user: root
  tasks:
    - name: install {{ item }} package
      yum: name={{ item }} state=latest
      with_items:
        - nginx
        - tomcat
        - mariadb-server
        - redis
[root@centos7 playbooks]#ansible-playbook --syntax-check 8.yaml
[root@centos7 playbooks]#ansible-playbook -C 8.yaml
```

8.角色(roles):

每个角色，以特定的层级目录结构进行组织：

如：nginx/

- files/：存放由copy或script模块等调用的文件；
- templates/：template模块查找所需要模板文件的目录；
- tasks/：至少应该包含一个名为main.yml的文件；其它的文件需要在此文件中通过include进行包含；
- handlers/：至少应该包含一个名为main.yml的文件；其它的文件需要在此文件中通过include进行包含；
- vars/：至少应该包含一个名为main.yml的文件；其它的文件需要在此文件中通过include进行包含；
- meta/：至少应该包含一个名为main.yml的文件，定义当前角色的特殊设定及其依赖关系；其它的文件需
- default/：设定默认变量时使用此目录中的main.yml文件；

在playbook调用角色方法1：

```
- hosts: webservs
  remote_user: root
  roles:
    - mysql
    - memcached
    - nginx
```

在playbook调用角色方法2：传递变量给角

```
- hosts:
  remote_user: root
  roles:
    - { role: nginx, username: nginx }
```

键role用于指定角色名称；后续的k/v用于传递变量给角色；

安装Nginx：




```
[root@centos7 ~]#vim /root/nginx.yml #角色配置文件
- hosts: webservs
  remote_user: root
  roles:
    - nginx
[root@centos7 ~]#ansible-playbook --syntax-check /root/nginx.yml
[root@centos7 ~]#ansible-playbook -C /root/nginx.yml
提供配置文件等:
[root@centos7 roles]#vim nginx/tasks/main.yml
- name: install nginx
  yum: name=nginx state=latest
  when: ansible_os_family == "RedHat"
[root@centos7 roles]#vim nginx/templates/vhost1.conf.j2
server {
    listen 80;
    server_name {{ ansible_fqdn }};
    location / {
        root "/ngxdata/vhost1";

[root@centos7 roles]#vim nginx/tasks/main.yml
- name: install nginx
  yum: name=nginx state=latest
  when: ansible_os_family == "RedHat"
- name: install conf
  template: src=vhost1.conf.j2 dest=/etc/nginx/conf.d/vhost1.conf
  tags: conf
  notify: restart nginx
- name: install site home directory
  file: path={{ ngxroot }} state=directory
- name: install index page
  copy: src=index.html dest={{ ngxroot }}/
- name: start nginx
  service: name=nginx state=started
[root@centos7 roles]#vim nginx/handlers/main.yml
- name: restart nginx
  service: name=nginx state=restarted
[root@centos7 roles]#vim nginx/vars/main.yml
ngxroot: /ngxdata/vhost1 #变量字典不需加-
[root@centos7 roles]#vim nginx/files/index.html
Nginx page
[root@centos7 playbooks]#ansible-playbook -C /root/nginx.yml
[root@centos7 playbooks]#ansible-playbook /root/nginx.yml
```

(/apps/redi
utm_sourc
banner-clic

小礼物走一走，来简书关注我

赞赏支持

 Linux (/nb/12772563)

举报文章 © 著作权归作者所有



jie0112 (/u/6800627be4d5)

写了 33542 字，被 13 人关注，获得了 16 个喜欢
(/u/6800627be4d5)

+ 关注

喜欢



更多分享

