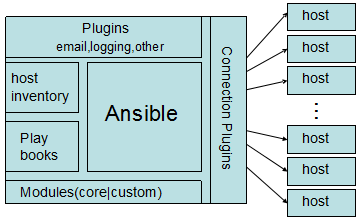
# Ansible

## 1、Ansible简介

1. 几种常用运维工具比较  
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  
SaltStack  
—基于 Python 开发,采用 C/S 架构,相对 puppet 更轻量级,配置语法使用 YAML,使得配置脚本更简单  
Ansible  
—基于 Python paramiko 开发,分布式,无需客户端,轻量级,配置语法使用YAML 及 Jinja2  
模板语言,更强的远程命令执行操作  
2. Ansible简介  
Ansible 基于Python开发,集合了众多运维工具（puppet、cfengine、chef、func、fabric）的优点,实现了批量系统配置、批量程序部署、批量运行命令等功能。ansible是基于模块工作的,本身没有批量部署的能力。真正具有批量部署的是ansible所运行的模块,ansible只是提供一种框架。  
  
connection plugins：连接插件,负责和被管理端实现通信,有SSH,ZEROMQ等,默认使用SSH连接  
host inventory：主机清单,是一个配置文件里面定义监控的主机  
modules : 模块,核心模块、command模块、自定义模块等  
plugins : modules功能的补充,包括连接插件,邮件插件等  
playbook：编排,定义 Ansible 多任务配置文件,非必需  
3. Ansible特性  
1)、no agents：不需要在被管控主机上安装任何客户端,更新时,只需在操作机上进行一次更新即可  
2)、no server：无服务器端,使用时直接运行命令即可  
3)、modules in any languages：基于模块工作,可使用任意语言开发模块  
4)、yaml,not code：使用yaml语言定制剧本playbook  
5)、ssh by default：基于SSH工作  
6)、strong multi-tier solution：可实现多级指挥  
  
4. Ansible工作过程:  
在ANSIBLE 管理体系中，存在"管理节点" 和 "被管理节点" 两种角色。被管理节点通常被称为"资产".在管理节点上，Ansible将 AdHoc 或 PlayBook 转换为Python脚本,并通过SSH将这些Python 脚本传递到被管理服务器上。在被管理服务器上依次执行，并实时的将结果返回给管理节点。

## 2、Ansible的安装

安装  
1. 配置EPEL网络yum源  
2. 安装ansible  
# yum install ansible -y  
3. 配置文件  
/etc/ansible/ansible.cfg #主配置文件,主要设置一些ansible初始化的信息，比如日志存放路径、模块、插件等配置信息  
/etc/ansible/hosts #主机清单文件,定义所管理的主机组及主机,可以在主配置文件中修改/usr/bin/ansible #Ansibe AD-Hoc 临时命令执行工具，常用于临时命令的执行 /usr/bin/ansible-doc # ansible 模块功能查看工具

/usr/bin/ansible-galaxy # 下载/上传优秀代码或Roles模块 的官网平台，基于网络的

/usr/bin/ansible-playbook # ansible 定制自动化的任务集编排工具

/usr/bin/ansible-pull # ansible远程执行命令的工具，拉取配置而非推送配置（使用较少，海量机器时使用，对运维的架构能力要求较高）

/usr/bin/ansible-vault # ansible 文件加密工具

/usr/bin/ansible-console # ansible基于Linux Consoble界面可与用户交互的命令执行工具

## 3、Ansible命令详解

###命令格式：

ansible <host-pattern> [-f forks] [-m module\_name] [-a args]

[root@CentOS7-master ~]# ansible -h

Usage: ansible <host-pattern> [options] Options:

\*\* -a MODULE\_ARGS, --args=MODULE\_ARGS模块的参数,如果执行默认COMMAND的模块，即是命令参数,如：“date”,“pwd”等等 module arguments 模块参数

-k, --ask-pass

ask for SSH password 登录密码，提示输入SSH密码而不是假设基于密钥的验证

--ask-su-pass

ask for su password su切换密码

-K, --ask-sudo-pass

ask for sudo password 提示密码使用sudo,sudo表示提权操作

--ask-vault-pass

ask for vault password

-B SECONDS, --background=SECONDS

run asynchronously, failing after X seconds (default=N/A)后台运行超时时间

\*\* -C, --check don‘t make any changes; instead, try to predict some of the changes that may occur #只是测试一下会改变什么内容，不会真正去执行;相反,试图预测一些可能发生的变化

-c CONNECTION, --connection=CONNECTION connection type to use (default=smart) #连接类型使用

\*\* -f FORKS, --forks=FORKS specify number of parallel processes to use (default=5) #并行任务数。NUM被指定为一个整数,默认是5

-h, --help show this help message and exit 打开帮助文档API

\*\* -i INVENTORY, --inventory-file=INVENTORY specify inventory host file (default=/etc/ansible/hosts) #指定库存主机文件的路径,默认为/etc/ansible/hosts

\*\* --list-hosts outputs a list of matching hosts; does not execute anything else

\*\* -m MODULE\_NAME, --module-name=MODULE\_NAME module name to execute (default=command) #执行模块的名字，默认使用 command 模块，所以如果是只执行单一命令可以不用 -m参数

-M MODULE\_PATH, --module-path=MODULE\_PATH specify path(s) to module library (default=/usr/share/ansible/) #要执行的模块的路径，默认为/usr/share/ansible/ -o, --one-line condense output #压缩输出，摘要输出.尝试一切都在一行上输出。

-P POLL\_INTERVAL, --poll=POLL\_INTERVAL set the poll interval if using -B (default=15) #调查背景工作每隔数秒。需- b

--private-key=PRIVATE\_KEY\_FILE use this file to authenticate the connection #私钥路径，使用这个文件来验证连接

\*\* -S, --su run operations with su #用 su 命令

\*\* -R SU\_USER, --su-user=SU\_USER run operations with su as this user (default=root) #指定SU的用户，默认是root用户

\*\* -s, --sudo run operations with sudo (nopasswd)

\*\* -U SUDO\_USER, --sudo-user=SUDO\_USER desired sudo user (default=root)(deprecated, use become) #sudo到哪个用户，默认为 root

\*\* -T TIMEOUT, --timeout=TIMEOUT override the SSH timeout in seconds (default=10) #指定SSH默认超时时间， 默认是10S

-t TREE, --tree=TREE log output to this directory #将日志内容保存在该输出目录,结果保存在一个文件中在每台主机上。

\*\* -u REMOTE\_USER, --user=REMOTE\_USER connect as this user (default=root) #远程用户， 默认是root用户

--vault-password-file=VAULT\_PASSWORD\_FILE vault password file

\*\* -v, --verbose verbose mode (-vvv for more, -vvvv to enable 详细信息connection debugging) --version show program's version number and exit #输出ansible的版本

## 4、Ansible命令举例

Ansible 默认提供了很多模块来供我们使用。在 Linux 中，我们可以通过 ansible-doc -l 命令查看到当前 ansible 都支持哪些模块，通过 ansible-doc -s 模块名 又可以查看该模块有哪些参数可以使用。

我们常用的几个模块：  
copy file cron group user yum service script ping command raw get\_url synchronize  
'host'：自己定义的主机 -m command：命令  
#在指定节点上运行uptime指令  
ansible 'host' -m command -a 'uptime'  
# 指定节点上的权限，属主和数组为root  
ansible 'host' -m file -a "dest=/tmp/t.sh mode=755 owner=root group=root"  
#指定节点上定义一个计划任务，每隔3分钟到主控端更新一次时间  
ansible 'host' -m cron -a 'name="custom job" minute=host/3 hour=host day=host month=host weekday=host job="/usr/sbin/ntpdate 172.16.254.139"'  
# 指定节点上创建一个组名为aaa，gid为2017的组  
ansible all -m group -a 'gid=2017 name=a'  
# 在节点上创建一个用户aaa，组为aaa  
ansible all -m user -a 'name=aaa groups=aaa state=present'  
#删除用户示例  
ansible all -m user -a 'name=aaa groups=aaa remove=yes'  
# 在节点上安装httpd  
ansible all -m yum -a "state=present name=httpd"  
# 在节点上启动服务，并开机自启动  
ansible all -m service -a 'name=httpd state=started enabled=yes'  
# 检查主机连接  
ansible 'host' -m ping  
# 执行远程命令  
ansible 'host' -m command -a 'uptime'  
# 执行主控端脚本  
ansible 'host' -m script -a '/root/test.sh'  
# 执行远程主机的脚本  
ansible 'host' -m shell -a 'ps aux|grep zabbix'  
# 类似shell  
ansible 'host' -m raw -a "ps aux|grep zabbix|awk '{print \$2}'"  
# 创建软链接  
ansible 'host' -m file -a "src=/etc/resolv.conf dest=/tmp/resolv.conf state=link"  
  
# 删除软链接  
ansible 'host' -m file -a "path=/tmp/resolv.conf state=absent"  
# 复制文件到远程服务器  
ansible 'host' -m copy -a "src=/etc/ansible/ansible.cfg dest=/tmp/ansible.cfg owner=root group=root mode=0644"  
# 在节点上运行hostname  
nsible all -m raw -a 'hostname|tee'  
# 将指定url上的文件下载到/tmp下  
ansible all -m get\_url -a 'url=http://10.1.1.116/favicon.ico dest=/tmp'  
ynchronize模块：  
目的：将主控方/root/a目录推送到指定节点的/tmp目录下  
命令：ansible all -m synchronize -a 'src=/root/a dest=/tmp/ compress=yes'  
执行效果：  
delete=yes 使两边的内容一样（即以推送方为主）  
compress=yes 开启压缩，默认为开启  
--exclude=.git 忽略同步.git结尾的文件  
由于模块，默认都是推送push。因此，如果你在使用拉取pull功能的时候，可以参考如下来实现  
mode=pull 更改推送模式为拉取模式  
目的：将10.1.1.113节点的/tmp/a目录拉取到主控节点的/root目录下  
命令：ansible 10.1.1.113 -m synchronize -a 'mode=pull src=/tmp/a dest=/root/'

## 5、Ansible组件-Inventory



Inventory 主机清单  
========================================================  
[http://docs.ansible.com/ansible/intro\_inventory.html#](http://docs.ansible.com/ansible/intro_inventory.html)  
  
inventory文件通常用于定义要管理主机的认证信息，例如ssh登录用户名、密码以及key相关信息。  
/etc/ansible/hosts [缺省]  
主机清单文件配置格式  
[webservers]  
192.168.10.128  
Bar.example.com  
up.example.com:5309 #指定 SSH 端口 5309  
web1 ansible\_ssh\_host=192.168.1.50 #设置主机别名为 web1  
www[01:50].example.com #支持通配符匹配www01,www02,...,www50  
db-[a:f].example.com #通配符匹配db-a,db-b,...,db-f   
-----------------------------------------------------------------  
为每个主机单独指定一些变量，这些变量可以在 playbooks 中使用：  
[atlanta]  
host1 http\_port=80 maxRequestsPerChild=808  
host2 http\_port=303 maxRequestsPerChild=909  
------------------------------------------------------------------  
为一个组指定变量，组内每个主机都可以使用该变量：  
[atlanta]  
host1  
host2  
[atlanta:vars]  
ntp\_server=ntp.atlanta.example.com  
proxy=proxy.atlanta.example.com  
-----------------------------------------------------------------  
组可以包含其他组：  
[atlanta]  
host1  
host2  
[raleigh]  
host3  
host4  
[southeast:children]  
atlanta  
raleigh  
[southeast:vars]  
some\_server=foo.southeast.example.com  
halon\_system\_timeout=30  
hosts文件支持一些特定指令:

## 6、Ansible组件-Ad-Hoc

**Ad-Hoc**  
========================================================  
ad-hoc —— 临时的，在ansible中是指需要快速执行，并且不需要保存的命令。说白了就是执行  
简单的命令——一条命令。对于复杂的命令则为 playbook。  
  
基本格式:  
ansible <pattern> -m <module\_name> -a <arguments>  
pattern: inventory文件里定义的主机组名,主机名,IP,别名等,all表示所有的主机,支持通配符,正则  
: - 冒号,多个组,组名之间用冒号隔开  
\*web\* - 组名或主机名中含web的  
webservers[0] - webservers组中的第一台主机  
以~开头,匹配正则  
-m module\_name: 模块名称,默认为command  
-a arguments: 传递给模块的参数  
-i file\_name: 指定inventory文件,默认为/etc/ansible/hosts  
  
配置过程:  
1.在控制节点安装ansible   
2.在控制节点生成密钥,把公钥传到所有的被控制节点  
3.编辑主机清单文件,加入被管控节点  
  
帮助信息查看  
# ansible-doc -l 查看所有模块信息  
# ansible-doc module\_name 查看模块帮助信息

## 7、Ansible组件-Modules

**Modules**  
========================================================  
常用模块介绍:  
1. setup: 查看远程主机的基本信息 - Facts - 用于采集被管理主机信息  
收集可用的facts,收集每个节点的相关信息, 如: 架构信息，IP,时间，域名，网卡，MAC，主机名，CPU等信息。  
物理内存容量 ansible\_memtotal\_mb  
CPU型号 ansible\_processor  
CPU核心数 ansible\_processor\_cores  
操作系统类型 ansible\_os\_family  
操作系统内核 ansible\_kernel  
硬盘挂载名及容量 ansible\_mounts  
服务器主机名 ansible\_nodename   
服务器ipv4地址 ansible\_all\_ipv4\_addresses  
常用选项:  
filter: 仅列出匹配到的facts, 默认值为\*  
ansible all -m setup -a "filter=ansible\_os\_family"  
2. ping: 测试远程主机的运行状态

ansible 'host' -m ping  
3. file: 设置文件属性  
相关选项如下：  
force：需要在两种情况下强制创建软链接，一种是源文件不存在，但之后会建立的情况下；另一种是目标软链接已存在，需要先取消之前的软链，然后创建新的软链，有两个选项：yes|no  
group：定义文件/目录的属组  
mode：定义文件/目录的权限  
owner：定义文件/目录的属主  
path：必选项，定义文件/目录的路径  
recurse：递归设置文件的属性，只对目录有效，有两个选项：yes|no  
src：被链接的源文件路径，只应用于state=link的情况  
state：  
      directory：如果目录不存在，就创建目录  
      file：即使文件不存在，也不会被创建  
      link：创建软链接  
      hard：创建硬链接  
 touch：如果文件或目录已存在，则更新其最后修改时间  
       absent：删除文件  
4. copy: 把主控端的文件复制到远程主机  
相关选项如下：  
backup：在覆盖之前，将源文件备份，备份文件包含时间信息。有两个选项：yes|no  
content：用于替代“src”，可以直接设定指定文件的值  
dest：必选项。要将源文件复制到的远程主机的绝对路径，如果源文件是一个目录，那么该路径也必须是个目录  
directory\_mode：递归设定目录的权限，默认为系统默认权限  
force：如果目标主机包含该文件，但内容不同，如果设置为yes，则强制覆盖，如果为no，则只有当目标主机的目标位置不存在该文件时，才复制。默认为yes  
others：所有的file模块里的选项都可以在这里使用  
src：被复制到远程主机的本地文件，可以是绝对路径，也可以是相对路径。如果路径是一个目录，它将递归复制。在这种情况下，如果路径使用“/”来结尾，则只复制目录里的内容，如果没有使用“/”来结尾，则包含目录在内的整个内容全部复制，类似于rsync  
5. shell: shell命令  
ansible默认使用的模块是command,支持多数shell命令,但不支持某些shell变量及管道,如果要使用,用shell模块  
  
6. user: 用户管理  
相关选项：  
name: 要操作的用户名  
uid:设置用户的UID  
group:设置用户的基本组  
groups:设置用户的附加组,如果设置为空,则将用户从所有附加组中退出来  
shell:设置用户登录shell   
home:设置用户家目录  
state:  
present: 创建 默认值  
absent: 删除  
remove: 当state=absent时,使用remove=yes干净的删除,默认值为no  
  
7. cron: 计划任务管理  
相关选项：  
name: 要管理的计划任务条目名称  
minute: 分  
hour: 时  
day: 日  
month: 月  
weekday: 周  
job: 要执行的命令  
user: 用户  
state:   
present: 创建 默认值  
absent: 删除, 会列出现有的所有的任务  
8. script: 将本地脚本传输到远程执行  
相关选项：  
free\_form: 任意的本地可执行脚本的路径  
creates: 接一个文件名,如果此文件已存在,将不执行脚本  
removes: 接一个文件名,如果此文件不存在,将不执行脚本  
9. service: 服务管理(el7:systemd)  
相关选项：  
name: 服务名称  
state:  
started: 启动服务  
stopped: 停止服务  
restarted: 重启服务  
reloaded: 重新加载配置文件  
enabled: 是否开机启动 yes | no  
10. lineinfile: 文件编辑  
相关选项：  
﻿dest:指定要修改的文件  
    regexp:匹配要修改的内容

## 8、Ansible组件-PlayBook

**PlayBooks**  
========================================================  
<http://docs.ansible.com/ansible/YAMLSyntax.html>  
  
Playbooks 是 Ansible 管理配置、部署应用和编排的语言，可以使用 Playbooks 来描述你想在远程主机执行的策略或者执行的一组步骤过程等  
如果说 Ansible 模块是工作中的工具的话，那么 playbooks 就是方案  
  
Playbooks 采用 YAML 语法结构  
Playbooks 组成

parget section

定义将要执行 playbook 的远程主机组  
Variable section  
定义 playbook 运行时需要使用的变量  
Task section  
定义将要在远程主机上执行的任务列表  
Handler section  
定义 task 执行完成以后需要调用的任务  
  
简单示例1:  
[root@master ~]# cd /etc/ansible/  
[root@master ansible]# vim test.yml                             #后缀为yml  
---  
- hosts: testhost  
  user: root  
  tasks:  
    - name: playbook\_test  
      shell: touch /tmp/playbook.txt  
注意：  
hosts参数指定了对哪些主机进行操作；  
user参数指定了使用什么用户登录远程主机操作；  
tasks指定了一个任务，其下面的name参数同样是对任务的描述，在执行过程中会打印出来。  
[root@master ansible]# ansible-playbook test.yml   
  
简单示例2:  
[root@master ansible]# vim create\_user.yml  
---  
- name: create\_user  
  hosts: testhost  
  user: root  
  gather\_facts: false  
  vars:  
    - user: "uplooking"  
  tasks:  
    - name: create user  
      user: name="{{ user }}"  
注意：  
name参数对该playbook实现的功能做一个概述，后面执行过程中，会打印 name变量的值 ，可以省略；  
gather\_facts参数指定了在以下任务部分执行前，是否先执行setup模块获取主机相关信息，这在后面的task会使用到setup获取的信息时用到；  
vars参数指定了变量，这里指字一个user变量，其值为test ，需要注意的是，变量值一定要用引号引住；  
user提定了调用user模块，name是user模块里的一个参数，而增加的用户名字调用了上面user变量的值。  
[root@master ansible]# ansible-playbook create\_user.yml  
  
简单示例3:  
[root@master ansible]# vim when.yml  
---  
- hosts: testhost  
  user: root  
  gather\_facts: True  
  tasks:  
    - name: use when  
      shell: touch /tmp/when.txt  
      when: ansible\_fqdn == "clone2.up.com"  
  
注意：只有当参数 facter\_ipaddress 为 192.168.10.110 时才在该机器上新建指定文件；意思就是只对 testhost 组中特定的主机进行操作，忽略组内其他的主机。我们可以通过setup模块查看各个参数的值  
[root@master ansible]# ansible-playbook when.yml  
  
简单示例4:  
[root@master ansible]# vim handlers.yml  
---  
- name: handlers test  
  hosts: testhost  
  user: root  
  tasks:  
    - name: test copy  
      copy: src=/etc/passwd dest=/tmp/handlers.txt  
      notify: test handlers  
  handlers:  
    - name: test handlers  
      shell: echo "www.uplooking.com" >> /tmp/handlers.txt  
  
说明：只有 copy 模块真正执行后，才会去调用下面的 handlers 相关的操作，追加内容。所以这种比较适合配置文件发生更改后，需要重启服务的操作。  
[root@master ansible]# ansible-playbook handlers.yml

## 9、Ansible组件-Roles

**Roles**  
==================================================  
"ansible all -i /app/ansible-playbook/hosts -m ping” 这种执行方式被称为ad-hoc模式，即命令行或交互模式  
但任何配置管理工具的官方文档都会告诉你要用编排的方式进行复杂的部署，例如saltstack里的.sls文件，ansible里的playbook。  
  
除此之外，ansible提供了一种目录树结构的编排方式，顶层目录对应roles，里面包含子目录，比如defaults、files、tasks、templates等  
不同的子目录对应不同的功能，这种方式使得管理和重复调用变得极为方便。  
例:





