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

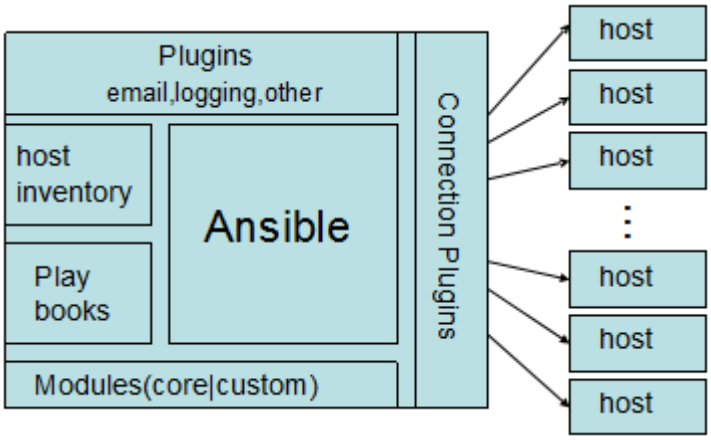
—基于 Python 开发,采用 C/S 架构,相对 puppet 更轻量级,配置语法使用 YAML,使得配置脚本更简单

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

3、ANSIBLE 常见命令

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/'

4、ANSIBLE 组件-INVENTORY

参数	用途	例子
ansible_ssh_host	定义 hosts ssh 地址	ansible_ssh_host=192.169.1.100
ansible_ssh_port	定义 hosts ssh 端口	ansible_ssh_port=3000
ansible_ssh_user	定义 hosts ssh 认证用户	ansible_ssh_user=user
ansible_ssh_pass	定义 hosts ssh 认证密码	ansible_ssh_pass=pass
ansible_sudo	定义 hosts sudo 用户	ansible_sudo=www
ansible_sudo_pass	定义 hosts sudo 密码	ansible_sudo_pass=pass
ansible_sudo_exe	定义 hosts sudo 路径	ansible_sudo_exe=/usr/bin/sudo
ansible_connection	定义 hosts 连接方式	ansible_connection=local
ansible_ssh_private_key_file	定义 hosts 私钥	ansible_ssh_private_key_file=/root/key
ansible_ssh_shell_type	定义 hosts shell 类型	ansible_ssh_shell_type=bash
ansible_python_interpreter	定义 hosts 任务执行python路径	ansible_python_interpreter=/usr/bin/python2.6
ansible_*_interpreter	定义 hosts 其它语言解析路径	ansible_*_interpreter=/usr/bin/ruby

Inventory 主机清单

==========

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：

5、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

6、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

7、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
```

8、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:

用ansible编译安装nginx

注意：

1.roles下子目录里必须要有main.yml文件，ansible会自动查询并执行。

2.roles目录和nginx.yml放在同一级目录中，或者在ansible.cfg中配置roles的查询路径。

3.如果yml中冒号后引用jinja模板以{{开头，则整行语句需要加上" "，防止yml认为这是个字典。

```
[root@localhost app]# tree ansible-playbook
ansible-playbook
├── nginx.yml
└── roles
    └── nginx                    #这就是在nginx.yml主文件中指定的role
        ├── defaults
        │   └── main.yml
        ├── files
        │   ├── compile.sh.j2
        │   └── nginx-1.6.3.tar.gz
        ├── handlers
        │   └── main.yml
        ├── tasks
        │   ├── install.yml
        │   └── main.yml
        └── templates
            └── nginx.conf.j2

1.defaults中存放默认的变量，可以通过jinja模板调用
2.files中存放文件、软件包、脚本等内容，可以被copy、unarchive、script等模块调用
3.handlers中存放依赖任务，可以被notify关键字调用
4.tasks中存放主任务，ansible会首先进行调用
5.templates中存放模板文件，模板中可以使用jinja模板调用defaults中定义的变量，被templates模块调用
```

defaults

```
[root@localhost nginx]# tree defaults
defaults
└── main.yml

[root@localhost nginx]# less defaults/main.yml

user: nginx
group: nginx
tarball_name: nginx-1.6.3.tar.gz
nginx_configuration: nginx.conf.j2
nginx_dir: nginx-1.6.3
nginx_port: 2223              #这是我们刚才在模板文件中使用的变量

#defaults中的变量优先级最低，通常我们可以临时指定变量来进行覆盖
```

tasks

nginx的安装过程包括创建用户、创建目录、下载安装包、下载依赖包、编译安装、创建软链接、修改配置文件、测试、启动这些环节。

```
[root@localhost nginx]# tree tasks/
tasks/
├── install.yml
└── main.yml

[root@localhost nginx]# less tasks/main.yml

---
- import_tasks: install.yml

#ansible的playbook以---开头，然后使用yml语法编写
#tasks/main.yml中加载install.yml，include方式不久会被淘汰，这里采用import_tasks关键字

[root@localhost nginx]# less tasks/install.yml

---
- name: groupadd nginx   #创建组，存在则忽略，group模块   - name:说明
  group:
    name: "{{ group }}"
    gid: 888

- name: useradd nginx   #创建用户，存在则忽略，user模块
  user:
    name: "{{ user }}"
    group: "{{ group }}"
    uid: 888
    createhome: no
    shell: /sbin/nologin

- name: install pcre-devel  #安装依赖，package模块
  package:
    name: pcre-devel
    state: latest

```

```
- name: install openssl-devel  #安装依赖,package模块
  package:
    name: openssl-devel
    state: latest

- name: create /tools    #创建目录,file模块
  file:
    path: /tools
    state: directory

- name: copy and extract nginx tarball  #解压压缩包，unarchive模块
  unarchive:
    src: "{{ tarball_name }}"
    dest: /tools

- name: ./configure         #检查环境，command模块
  command: ./configure --user={{ user }} --group={{ group }} --prefix=/app/{{ nginx_dir }} --with-http_stub_s
tatus_module --with-http_ssl_module
  args:
    chdir: /tools/{{ nginx_dir }}

- name: make    #编译，command模块
  command: make
  args:
    chdir: /tools/{{ nginx_dir }}

- name: make install    #安装，command模块
  command: make install
  args:
    chdir: /tools/{{ nginx_dir }}

- name: modify nginx configuration   #修改配置文件，template模块
  template:
    src: "{{ nginx_configuration }}"
    dest: /app/{{ nginx_dir }}/conf/nginx.conf

- name: make link     #创建软连接，file模块
  file:
    src: /app/{{ nginx_dir }}
    dest: /app/nginx
    state: link

- name: test nginx   #测试nginx配置,command模块
  command: /app/nginx/sbin/nginx -t
  notify:              #调用handlers目录下的main.yml
```

handlers

```
[root@localhost nginx]# tree handlers/
handlers/
└── main.yml

[root@localhost nginx]# less handlers/main.yml

---
- name: start nginx     #notify下面指定的内容在name这里定义
  command: /app/nginx/sbin/nginx

#这里只是演示，实际批量部署不需要nginx -t 这一步
```

files

```
[root@localhost nginx]# tree files/
files/
└── nginx-1.6.3.tar.gz

#ansible中unarchive、copy等模块会自动来这里找文件，从而我们不必写绝对路径，只需写文件名
```

templates

```
[root@localhost nginx]# tree templates/
templates/
└── nginx.conf.j2

#一般来说，模板文件中都会使用jinja2模板，所以通常我们在模板文件后加上.j2后缀，但不是必须的

[root@localhost nginx]# less templates/nginx.conf.j2

    server {
        listen       {{ nginx_port }};    #这里使用jinja模板引用变量
        server_name  localhost;


#template模块会将模板文件中的变量替换为实际值，然后覆盖到客户机指定路径上
```

执行playbook

到了激动人心的时刻，ansible的好处在于什么都不用配，直接就能用，所以这里我们将inventory、nginx.yml、roles目录放在同一级目录ansible-playbook下，便于管理

```
#首先看看nginx.yml主文件

[root@localhost ansible-playbook]# less nginx.yml

---
- name: deploy nginx
  hosts: all
  remote_user: root
  roles:
    - nginx


#hosts表示选择哪些主机进行部署
#remote_user表示选择哪个用户进行部署
#roles表示选择部署什么内容

#当然，这里还可以通过字典的方式指定不同的变量
---
- name: deploy nginx
  hosts: all
  remote_user: root
  roles:
    - { role: nginx, nginx_port: 8080 }
```

我们进入ansible-playbook目录下，执行 ansible-playbook -i hosts nginx.yml 即可开始部署

```
[root@localhost ansible-playbook]# ansible-playbook -i hosts nginx.yml

TASK [Gathering Facts]   ********************************************************************************************
ok: [172.16.1.10]

TASK [nginx : groupadd nginx]  ***************************************************************************************
ok: [172.16.1.10]

TASK [nginx : useradd nginx] *****************************************************************************************
ok: [172.16.1.10]

。。。。。

# TASK[]里的内容就是定义在首行name中的提示内容
# -i 表示自行指定inventory文件
```