单元三 ansbile介绍

## 1.Ansible简介

参考资料地址：

* 官网

https://www.ansible.com/

* Ansible介绍视频

https://www.ansible.com/resources/videos/quick-start-video

* Ansible中文权威指南

http://ansible-tran.readthedocs.io/en/latest/

* Ansible自动化运维教程

https://www.w3cschool.cn/automate\_with\_ansible/

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：

让我们自动化部署APP；

自动化管理配置项；

自动化的持续交付；

自动化的（AWS）云服务管理。

所有的这几个目标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,批量的在远程服务器上执行命令 。

ansible是一个轻量级的运维自动化配置管理和配置工具，基于Python研发。集合了众多老牌运维工具（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。

这是一款很简单也很容易入门的部署工具，它使用SSH连接到服务器并运行配置好的任务，服务器上不用安装任何多余的软件，只需要开启ssh，所有工作都交给client端的ansible负责。因为基于SSH，所以不需要像Salt这些同类工具一样需要在每台机器上安装agent，因此添加机器几乎不需要进行额外的操作。仅需在管理工作站上安装ansible程序配置被管控主机的IP信息，被管控的主机无客户端。ansible应用程序存在于epel(第三方社区)源，依赖于很多python组件。ansible基于 paramiko 开发的,它是一个纯Python实现的ssh协议库。

主要包括：

(1)、连接插件connection plugins：负责和被监控端实现通信；

(2)、host inventory：指定操作的主机，是一个配置文件里面定义监控的主机；

(3)、各种模块：核心模块、command模块、自定义模块；

(4)、借助于插件完成记录日志、邮件等功能；

(5)、playbook：剧本执行多个任务时，非必需可以让节点一次性运行多个任务。

关于Ansible的一个好处是，将bash脚本转换为可执行任务是非常容易的。

我们可以使用shell编写自己的配置程序，但是Ansible更加干净，因为它可以自动在执行任务之前获取上下文。

ansible任务是幂等的，没有大量额外的编码，ansible可以一次又一次地安全运行bash命令，任意多次执行所产生的影响均与一次执行的影响相同，不会造成环境混乱。

ansible使用“facts”来确保任务的幂等安全运行， 它是在运行任务之前收集的系统和环境信息。ansible使用这些facts来检查状态，看看是否需要改变某些东西以获得所需的结果。这使得ansible可以让服务器一次又一次地运行可复制的任务。

## 2.ansible特性

一、模块化设计，调用特定的模块来完成特定任务，本身是核心组件，短小精悍；

二、基于Python语言实现，由Paramiko(python的一个可并发连接ssh主机功能库), PyYAML和Jinja2(模板化)三个关键模块实现；

三、部署简单，agentless无客户端工具；

四、主从模式工作；

五、支持自定义模块功能；

六、支持playbook剧本，连续任务按先后设置顺序完成；

七、期望每个命令具有幂等性：

## 3.ansible架构和核心组件

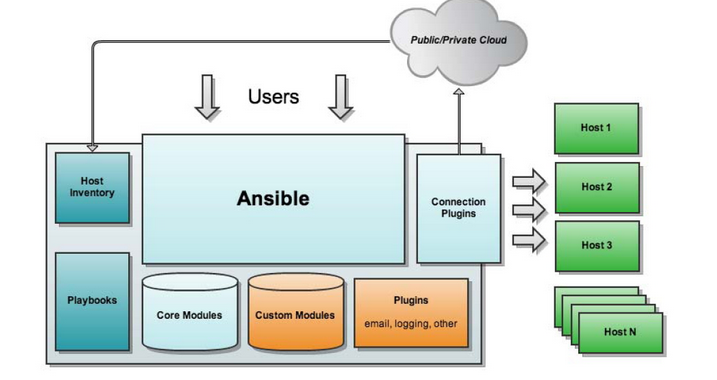


图3-1 ansible架构

核心组件：

* Host Inventory

定义ansible管理的主机，可以进行分组管理。

* Core Modules

Ansible核心模块，ansible中模块就是用来指定对远程主机具体的操作，比如执行命令模块command、创建文件模块file等（ansible自带的模块）。

* Custom Modules

自定义模块，如果核心模块不足以完成某种功能就可以使用任何语言自定义模块。

* Connection Plugins

连接插件是Ansible用来连接被管理端的一种方式，比如客户端运行了SSH服务，Ansible利用SSH服务跟客户端进行通信，Ansible是通过Python编写的，而Python有一个模块paramiko支持并行连接SSH。Ansible事实上就是使用paramiko进行连接被管理端。但是它还支持其他的连接方法需要插件（如zeroMQ就是C/S模式工作）。

* Plugins

第三方插件支持，如email、logging模块，只要有Python编写能力就可以自行编写插件。

* Playbooks

剧本或编辑是Ansible的任务配置文件、将多个任务定义在剧本中由Ansible自动执行，Playbooks最大的好处就是支持幂等，也就是说相同的任务不管执行多少次其结果是不变的，带来的好处也就是持续可维护性。

对管理主机的要求

目前,只要机器上安装了 Python 2.6 或 Python 2.7 (windows系统不可以做控制主机),都可以运行Ansible.

主机的系统可以是 Red Hat, Debian, CentOS, OS X, BSD的各种版本,等等.

自2.0版本开始,ansible使用了更多句柄来管理它的子进程,对于OS X系统,你需要增加ulimit值才能使用15个以上子进程,方法 sudo launchctl limit maxfiles 1024 2048,否则你可能会看见”Too many open file”的错误提示.

# 单元四 ansible安装

## 1.管理主机（master）安装ansible

安装Ansible之后,不需要启动或运行一个后台进程,或是添加一个数据库.只要在一台电脑(可以是一台笔记本)上安装好,就可以通过这台电脑管理一组远程的机器.在远程被管理的机器上,不需要安装运行任何软件,因此升级Ansible版本不会有太多问题(如果你已经基于Ansible开发大量模块，你最好一直使用对应版本。此时不建议你升级到最新版本，以免由于不兼容等问题导致模块功能异常。).

通常大家都喜欢用各个系统自带的包管理工具去安装和维护软件包。但这样，你并不一定能获取到最新或最可靠的Ansible版本。所以如果你想使用最新版本，最好使用pip来安装和管理Ansible。

Pip是专门用来管理Python模块的工具，Ansible会将每次正式发布都更新到pip仓库中。所以通过pip安装或更新Ansible，会比较稳妥的拿到最新稳定版。需要注意的是安装pip之前要先安装setuptools工具，使用setuptools工具带的命令easy\_install去安装pip，具体可以看python部分。

第一种：使用yum安装Ansible

使用yum安装Ansible时需要配置epel源才行，能帮我们自动解决软件包的依赖关系。

注意，非root用户需要sudo权限执行以下命令

|  |
| --- |
| $ yum install epel-release -y  $ yum install ansible -y |

过程如下：

|  |
| --- |
|  |

第二种：使用pip安装Ansible

如果使用pip安装Ansible。升级操作系统时，并不会同时升级Ansible。另外，升级操作系统有可能损坏Ansible环境，毕竟它依赖Python。Pip的安装指令为：

|  |
| --- |
| $ yum install epel-release -y  $ yum install python-pip  $ pip install ansible |

安装过程截图如下:

|  |
| --- |
|  |

第三种：使用源码安装Ansible

最时尚的玩法是使用源码安装了。你会拿到最新版，但并非稳定版。所以，使用源码安装时要留意Bug，积极关注社区和版本更新。请从Github上获取最新代码，安装过程如下：

|  |
| --- |
| $ yum install epel-release -y  $ yum install python-pip  $ pip install --upgrade setuptools  $ yum install git -y  $ git clone git://github.com/ansible/ansible.git  $ cd ansible  $ python setup.py install |

## 2.Ansible文件说明

通过以下命令可以查看一下Ansible相关文件。

|  |
| --- |
| $ rpm -ql ansible | more |

ansible的相关配置文件如下：

（1）# 主配置文件

/etc/ansible/ansible.cfg

（2）# 默认定义主机清单文件

/etc/ansible/hosts

（3）# 用来编排Playbook

/etc/ansible/roles

（4）# 执行命令的程序

/usr/bin/ansible

/usr/bin/ansible-console

（5）# Ansible模块使用帮助命令,其中使用-l可以查看ansible自带的所有模块

/usr/bin/ansible-doc

/usr/bin/ansible-galaxy

（6）# 用来执行playbook的程序

/usr/bin/ansible-playbook

/usr/bin/ansible-pull

/usr/bin/ansible-vault

安装过程中可能遇到问题：

Error: Cannot retrieve metalink for repository: epel. Please verify its path and try again

解决方法:

vi /etc/yum.repos.d/epel.repo

将[epel]下的mirrolist注释，baseurl放出了即可。

编辑[epel]下的baseurl前的#号去掉，metalink前添加#号。正确配置如下：

[epel]

name=Extra Packages for Enterprise Linux 7 - $basearch

baseurl=http://download.fedoraproject.org/pub/epel/7/$basearch

#metalink=https://mirrors.fedoraproject.org/metalink?repo=epel-7&arch=$basearch

failovermethod=priority

enabled=1

gpgcheck=1

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7

然后清理一下缓存

yum clean all

# 单元五 Inventory文件使用

既然Ansible是无agent模式的，那么就需要维护一份主机/分组的数据。Ansible只能管理指定的服务器，在inventory文件中进行配置对应的主机/分组的数据。

Inventory文件用来定义你要管理的主机，其默认位置在/etc/ansible/hosts，

如果不保存在默认位置，通过以下两种方式指定：

（1）可通过-i选项指定,

（2）可以通过修改ansible.cfg的inventory参数指定路径。

被管理的机器可以通过其IP或域名指定。每个中括号里代表一个分组，其下的机器列表都归属于这个分组，直到出现下一个中括号为止。通常我们按组来执行任务，同一组受控服务器应用相同的配置。一台服务器也可以归属到多个组，以完成多个功能角色的配置。低耦合、模块化，非常灵活！

如下定义一个分组[test]，并给定三台主机。

使用命令

|  |
| --- |
| [root@localhost ansible]# vi /etc/ansible/hosts |

在最后增加内容：

|  |
| --- |
| [test]  192.168.130.129  192.168.130.128 |

ansible命令用法：

$ ansible -i /etc/ansible/host all [options]

其中-i用来指定inventory文件，默认就是使用/etc/ansible/hosts，

其中all是针对hosts定义的所有主机执行，这里也可以指定hosts中定义的组名或模式。

[options]

-m：指定模块(默认是command模块)

-a：指定模块的参数

-u：指定执行远程主机的用户(默认是root)ansible.cfg中可配置

-k：指定远程主机的密码

-s：以sudo方式运行

-U：sudo到那个用户(默认是root)

-f：指定多少个进程并发处理(默认是5)

--private-key=/path：指定私钥路径

-T：ssh连接超时时间(默认(10s)

-t：日志输出到该目录

-v：显示详细信息

## 1.主机与组

### 1.1正常配置

/etc/ansible/hosts 文件的格式与windows的ini配置文件类似:

[webservers]

192.168.57.128

[dbservers]

192.168.57.128

方括号[]中是组名,用于对系统进行分类,便于对不同系统进行个别的管理.

执行如下命令：

|  |
| --- |
| [root@localhost ansible]# ansible webservers -m ping |

执行结果如下：

|  |
| --- |
|  |

一个系统可以属于不同的组,比如一台服务器可以同时属于 webserver组 和 dbserver组.，修改/etc/ansible/hosts文件如下：

|  |
| --- |
| [webservers]  192.168.57.128  192.168.57.129  [dbservers]  192.168.57.129 |

再次执行如下命令：

|  |
| --- |
| [root@localhost ansible]# ansible webservers -m ping |

执行结果如下：

|  |
| --- |
|  |

执行如下命令：

|  |
| --- |
| [root@localhost ansible]# ansible **dbservers** -m ping |

执行结果如下：

|  |
| --- |
|  |

### 1.2 SSH端口非默认

如果有主机的SSH端口不是标准的22端口,可在主机名之后加上端口号,用冒号分隔.SSH 配置文件中列出的端口号不会在 paramiko 连接中使用,会在 openssh 连接中使用.

注：修改SSH默认端口

|  |
| --- |
| [root@localhost ~]# vi /etc/ssh/sshd\_config |

修改如下内容：

|  |
| --- |
| Port 22 #把前面的注释去掉  **Port 2222 #新增一个端口** |

修改之后，进行保存。

**向防火墙中添加修改的端口**

向防火墙中添加端口的命令为:

|  |
| --- |
| [root@localhost ~]# firewall-cmd --zone=public --add-port=2222/tcp --permanent |

reaload 防火墙规则:

|  |
| --- |
| [root@localhost ~]# firewall-cmd --reload |

查看端口是否添加成功:

|  |
| --- |
| [root@localhost ~]# firewall-cmd --zone=public --query-port=2222/tcp |

**向SELinux中添加修改的SSH端口**

在向SELinux中添加端口之前需要先安装SELinux的管理工具 semanage (如果已经安装了就直接到下一步) : yum provides semanage

在这里我们看到运行 semanage 需要先安装其依赖工具包 policycoreutils-python :

|  |
| --- |
| [root@localhost ~]# yum install policycoreutils-python -y |

安装好之后，可以直接使用 semanage 命令:

查询当前 ssh 服务端口:

|  |
| --- |
| [root@localhost ~]# semanage port -l | grep ssh |

向 SELinux 中添加 ssh 端口:

|  |
| --- |
| [root@localhost ~]# semanage port -a -t ssh\_port\_t -p tcp 2222 |

验证 ssh 端口是否添加成功:

|  |
| --- |
| [root@localhost ~]# semanage port -l | grep ssh |

添加成功之后就可以重启 ssh 服务了:

|  |
| --- |
| [root@localhost ~]# systemctl restart sshd.service |

**到此就完成了端口的修改。**

端口号不是默认设置时,可明确的表示为:

修改/etc/ansible/hosts文件

|  |
| --- |
| 192.168.57.128:2222 |

### 1.3 设置别名

假设你有一些静态IP地址,希望设置一些别名,但不是在系统的 host 文件中设置,又或者你是通过隧道在连接,那么可以在/etc/ansible/hosts中设置如下:

server1 ansible\_ssh\_port=22 ansible\_ssh\_host=192.168.57.128

在这个例子中,通过 “ceph1” 别名,会连接 192.168.1.50:23.记住,这是通过 inventory 文件的特性功能设置的变量. 一般而言,这不是设置变量(描述你的系统策略的变量)的最好方式.后面会说到这个问题.

执行如下命令

|  |
| --- |
| [root@localhost ~]# ansible server1 -m ping |

执行结果：

|  |
| --- |
|  |

### 1.4相似主机名合并

首先修改虚拟机主机名

|  |
| --- |
| [root@localhost ~]# hostnamectl set-hostname ansible1 |

修改/etc/hosts文件

|  |
| --- |
| [root@ansible1 ~]# vi /etc/hosts |

增加以下两行内容

|  |
| --- |
| 192.168.130.129 ansible1  192.168.130.128 ansible2 |

对于一组相似的 hostname , 可简写如下:

[webservers]

ansible[1:2]

以上是对于数字模式的，还可以定义字母范围的简写模式:

[webservers]

ansible[a:b]

### 1.5 定义连接类型

对于每一个 host,你还可以选择连接类型和连接用户名:

[webservers]

localhost ansible\_connection=local

所有以上讨论的对于 inventory 文件的设置是一种速记法,后面我们会讨论如何将这些设置保存为 ‘host\_vars’ 目录中的独立的文件.

### 1.6主机变量

前面已经提到过,分配变量给主机很容易做到,这些变量定义后可在 playbooks 中使用:

[webservers]

ansiblea ansible\_connection=ssh

ansibleb ansible\_connection=ssh

### 1.7组的变量

也可以定义属于整个组的变量:

[webservers]

ansiblea

ansibleb

[webservers:vars]

ansible\_connection=ssh

把一个组作为另一个组的子成员

可以把一个组作为另一个组的子成员,以及分配变量给整个组使用. 这些变量可以给 /usr/bin/ansible-playbook 使用,但不能给 /usr/bin/ansible 使用:

[webservers]

ansiblea

ansibleb

[dbserves:children]

webservers

[dbserves:vars]

ansible\_connection=ssh

## 4.分文件定义 Host 和 Group 变量

在 inventory 主文件中保存所有的变量并不是最佳的方式.还可以保存在独立的文件中,这些独立文件与 inventory 文件保持关联. 不同于 inventory 文件(INI 格式),这些独立文件的格式为 YAML.详见 YAML 语法 .

假设 inventory 文件的路径为:

/etc/ansible/hosts

假设有一个主机名为 ‘foosball’, 主机同时属于两个组,一个是 ‘databases’, 另一个是 ‘webservers’. 那么以下配置文件(YAML 格式)中的变量可以为 ‘foosball’ 主机所用.依次为 ‘databases’ 的组变量,’webservers’ 的组变量,’foosball’ 的主机变量:

/etc/ansible/group\_vars/databases

/etc/ansible/group\_vars/webservers

/etc/ansible/host\_vars/foosball

举例来说,假设你有一些主机,属于不同的数据中心,并依次进行划分.每一个数据中心使用一些不同的服务器.比如 ntp 服务器, database 服务器等等. 那么 ‘databases’ 这个组的组变量定义在文件 ‘/etc/ansible/group\_vars/databases’ 之中,可能类似这样:

ntp\_server: acme.example.org

database\_server: storage.example.org

这些定义变量的文件不是一定要存在,因为这是可选的特性.

还有更进一步的运用,你可以为一个主机,或一个组,创建一个目录,目录名就是主机名或组名.目录中的可以创建多个文件, 文件中的变量都会被读取为主机或组的变量.如下 ‘databases’ 组对应于 /etc/ansible/group\_vars/databases/ 目录,其下有两个文件 db\_settings 和 cluster\_settings, 其中分别设置不同的变量:

/etc/ansible/group\_vars/databases/db\_settings

/etc/ansible/group\_vars/databases/cluster\_settings

‘databases’ 组下的所有主机,都可以使用 ‘databases’ 组的变量.当变量变得太多时,分文件定义变量更方便我们进行管理和组织. 还有一个方式也可参考,详见 Ansible Vault 关于组变量的部分. 注意,分文件定义变量的方式只适用于 Ansible 1.4 及以上版本.

Tip: Ansible 1.2 及以上的版本中,group\_vars/ 和 host\_vars/ 目录可放在 inventory 目录下,或是 playbook 目录下. 如果两个目录下都存在,那么 playbook 目录下的配置会覆盖 inventory 目录的配置.

Tip: 把你的 inventory 文件 和 变量 放入 git repo 中,以便跟踪他们的更新,这是一种非常推荐的方式.

## 5.Inventory 参数的说明

如同前面提到的,通过设置下面的参数,可以控制 ansible 与远程主机的交互方式,其中一些我们已经讲到过:

ansible\_ssh\_host

将要连接的远程主机名.与你想要设定的主机的别名不同的话,可通过此变量设置.

ansible\_ssh\_port

ssh端口号.如果不是默认的端口号,通过此变量设置.

ansible\_ssh\_user

默认的 ssh 用户名

ansible\_ssh\_pass

ssh 密码(这种方式并不安全,强烈建议使用 --ask-pass 或 SSH 密钥)

ansible\_sudo\_pass

sudo 密码(这种方式并不安全,强烈建议使用 --ask-sudo-pass)

ansible\_sudo\_exe (new in version 1.8)

sudo 命令路径(适用于1.8及以上版本)

ansible\_connection

与主机的连接类型.比如:local, ssh 或者 paramiko. Ansible 1.2 以前默认使用 paramiko.1.2 以后默认使用 'smart','smart' 方式会根据是否支持 ControlPersist, 来判断'ssh' 方式是否可行.

ansible\_ssh\_private\_key\_file

ssh 使用的私钥文件.适用于有多个密钥,而你不想使用 SSH 代理的情况.

ansible\_shell\_type

目标系统的shell类型.默认情况下,命令的执行使用 'sh' 语法,可设置为 'csh' 或 'fish'.

ansible\_python\_interpreter

目标主机的 python 路径.适用于的情况: 系统中有多个 Python, 或者命令路径不是"/usr/bin/python",比如 \\*BSD, 或者 /usr/bin/python

不是 2.X 版本的 Python.[我们](http://my.525.life/article?id=1510739742099 "我们")不使用 "/usr/bin/env" 机制,因为这要求远程用户的路径设置正确,且要求 "python" 可执行程序名不可为 python以外的名字(实际有可能名为python26).

与 ansible\_python\_interpreter 的工作方式相同,可设定如 ruby 或 perl 的路径....

一个主机文件的例子:

some\_host ansible\_ssh\_port=2222 ansible\_ssh\_user=manager

aws\_host ansible\_ssh\_private\_key\_file=/home/example/.ssh/aws.pem

freebsd\_host ansible\_python\_interpreter=/usr/local/bin/python

ruby\_module\_host ansible\_ruby\_interpreter=/usr/bin/ruby.1.9.3

相关经验

可能有些场景下，执行配置管理需要用root权限。但由于安全原因，可能会限制root使用SSH登录。比如：Ubuntu系统默认就不能使用 root 直接SSH登录系统。Ansible设计时也考虑到此类场景，这种情况下我们只需要告诉Ansible使用sudo的方式执行那些需要 root 权限的配置任务。前提条件是执行Ansible任务的用户需要有sudo的权限。要设定Ansible使用sudo，执行Ansible的任务的用户必须要有sudo权限。可以通过修改/etc/sudoers文件或visudo命令来完成，或者直接使用现在符合条件的用户。参数-sudo即告诉Ansible使用sudo来运行任务，如果sudo需要密码，则需要添加-k参数，或者在配置文件/etc/ansible/ansible.cfg中添加ask\_sudo\_pass的属性。

Ansible使用键值方式接受参数，即传统的KV方式（key=value）。每次执行任务后，将以JSON格式返回结果。它可以解析复杂的参数，或者用playbooks方式（后面会讲解）。Ansible返回会明确指明运行是否成功、是否有变动以及失败时的错信息。

通常都用playbooks的方式来执行Ansible任务，少数情况下使用命令行模式运行。过去，我们用Ansible的ping模块来检查“受控节点”是否正常受控。而实际上，ping模块仅仅执行了Ansible最核心的功能并检查了网络联通性，并未做其它实际性动作。

进而产生了setup模块，它不仅可以反馈“受控节点”的可用性，还会收集一些系统信息以供其它模块使用。Setup 模块定义了一系列的采集指令，比如：内核版本、机器名、IP地址等等，并将这些信息保存在内置变量，其它模块再执行任务时可以直接引用或用于判断条件等。

## 6.配置文件

安装完成之后，先来配置下配置项——.ansible.cfg。

ansible执行的时候会按照以下顺序查找配置项:

\* ANSIBLE\_CONFIG (环境变量)

\* ansible.cfg (当前目录下)

\* .ansible.cfg (用户家目录下)

\* /etc/ansible/ansible.cfg

## 7.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。对于复杂的命令后面会说playbook。

尝试运行

配置好主机列表之后，就可以开始执行批量任务了。Ansible自带了ping模块，可以测试“管理节点”和和“受控节点”之间的网络联通情况。

Linux服务器均使用SSH进行远程管理，SSH服务是服务器必备组件。Ansible在设计上简化使用成本，在“管理节点”上使用SSH连接“受控节点”。通过创建SSH连接来发送指使并执行，达到配置管理的目的。这样避免了不同版本的操作系统、甚至不同发行版的操作系统下使用Agent带来的兼容性问题。

首先，我们来赶紧体验一下Ansible的强大魅力。使用Ansible来测试“受控节点”的网络联通情况：

$ ansible -i /etc/ansible/hosts test -u root -m ping -k

或

$ ansible test -u root -m ping -k

输入主机密码后，输出正确信息如下：

注意了，如果是第一次使用SSH连接agent端，那么就不能使用-k了，要先进行公钥保存确认。然后再使用-k进行连接。

paramiko: The authenticity of host '192.168.1.73' can't be established.

The ssh-rsa key fingerprint is 6cef8eea7f799e6c24e3cfedf3838654.

Are you sure you want to continue connecting (yes/no)?

如果你使用密钥方式登录SSH，去掉-k参数即可。可以为Ansible设置一个特定的用户，所有操作均以此用户来执行。甚至可以为每个“受控节点”设置各自不同的用户。

全局用户的设置见配置文件/etc/ansible/ansible.cfg，修改[defaults]段落里的remote\_user的值即可。也可以通过修改remote\_port 来定义Ansible去使用非标准的22/SSH 端口来进行连接和管理。在没有给“受控节点”或“受控组”进行特定设置时，Ansible将默认使用全局设置。

1. 遇到报错：FAILED => **Using** a SSH password instead of a key is not possible because **Host** **Key** checking is enabled and sshpass does not support this. **Please** add this host's fingerprint to your known\_hosts file to manage this host.

默认host\_key\_checking部分是注释的，通过取消该行的注释，同样也可以实现跳过 ssh 首次连接提示验证部分。

host\_key\_checking = False

# 单元六 ansible常用模块介绍

根据Ansible官方的分类，将模块分为核心模块和额外模块，代码托管地址：https://github.com/ansible

核心模块按功能分类为：云模块、命令模块、数据库模块、文件模块、资产模块、消息模块、监控模块、网络模块、通知模块、包管理模块、源码控制模块、系统模块、单元模块、web设施模块、windows模块等。

具体可以参看官方页面ansible-modules-core。这里从官方分类的模块里选择最常用的一些模块进行介绍，介绍之前我们先调整一下hosts文件，把默认用户和密码添加到主机中，这样就不需要指定-u及-k了。

[test]

192.168.1.71 ansible\_ssh\_user=zzq ansible\_ssh\_pass=12345

192.168.1.72 ansible\_ssh\_user=zzq ansible\_ssh\_pass=12345

192.168.1.73 ansible\_ssh\_user=zzq ansible\_ssh\_pass=12345

## 一、ping模块

测试主机是否是通的，用法很简单，不涉及参数。

|  |
| --- |
| $ ansible test -m ping |

执行结果截图：

|  |
| --- |
|  |

## 二、setup模块

setup模块，主要用于获取主机信息，在playbooks里经常会用到的一个参数。gather\_facts就与该模块相关，setup模块下经常使用的一个参数是filter参数，具体使用示例如下：

# 查看主机内存信息

|  |
| --- |
| $ ansible test -m setup -a 'filter=ansible\_\*\_mb' |

# 查看地接口为eno\*的网卡信息

|  |
| --- |
| $ ansible test -m setup -a 'filter=ansible\_eno\*' |

# 将所有主机的信息输入到/tmp/facts目录下，每台主机的信息输入到主机名文件中（/etc/ansible/hosts里的主机名）

|  |
| --- |
| $ ansible test -m setup --tree /tmp/facts |

## 三、file模块

file模块主要用于远程主机上的文件操作，具体使用示例如下：

# 创建一个软连接

|  |
| --- |
| $ ansible webservers -m file -a "src=/etc/fstab dest=/tmp/fstab state=link" |

# 删除一个文件

|  |
| --- |
| $ ansible webservers-m file -a "path=/tmp/fstab state=absent" |

# 创建一个文件

|  |
| --- |
| $ ansible test -m file -a "path=/tmp/test state=touch" |

常用参数：

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, required;

recurse #递归的设置文件的属性，只对目录有效;

src #要被链接的源文件的路径，只应用于state=link的情况;

dest #被链接到的路径，只应用于state=link的情况;

state：

directory #如果目录不存在，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 #如果文件不存在，则会创建一个新的文件，如果文件或目录已存在，则更新其最后修改时间;

absent #删除目录、文件或者取消链接文件;

## 四、copy模块

复制文件到远程主机，如下示例：

|  |
| --- |
| $ ansible all -m copy -a "src=/root/ansible/ansible.txt dest=/tmp/ansible owner=ansible group=ansible mode=0644" |

进行备份。

|  |
| --- |
| $ ansible all -m copy -a "src=/root/ansible/ansible.txt dest=/tmp/ansible owner=ansible group=ansible mode=0644 backup=yes" |

|  |
| --- |
| $ ansible test -m copy -a "src=/mine/sudoers dest=/etc/sudoers validate='visudo -cf %s'" |

常用参数：

backup #在覆盖之前将原文件备份，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;

directory\_mode #递归的设定目录的权限，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;

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, required;

src #要复制到远程主机的文件在本地的地址，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;

validate #复制文件前进行验证,文件的路径的验证是通过"%s";

## 五、service模块

用于管理主机服务，能够同时管理CentOS6和CentOS7，不区分CentOS6的service和CentOS7的systemctl，如下实例：如果没有安装http服务，需要先安装。

|  |
| --- |
| $ yum install httpd -y |

|  |
| --- |
| $ ansible test -m service -a "name=httpd state=started enabled=yes" |

|  |
| --- |
| $ ansible test -m service -a "name=httpd state=stopped" |

|  |
| --- |
| $ ansible test -m service -a "name=network state=restarted " |

常用参数：

name #必选项,服务名称;

state #对当前服务执行启动，停止、重启、重新加载等操作（started, stopped, restarted, reloaded）;

enabled #是否开机启动yes|no;

pattern #定义一个模式，如果通过status指令来查看服务的状态时，没有响应，就会通过ps指令在进程中根据该模式进行查找，如果匹配到，则认为该服务依然在运行;

runlevel #运行级别;

arguments #给命令行提供一些选项;

sleep #如果执行了restarted，在则stop和start之间沉睡几秒钟;

作业：

要求：用ftp服务验证service的使用方法

1. 验证一下ftp有没有安装，如果没有安装的话，进行安装(两台主机都需要）

|  |
| --- |
| $ yum install vsftpd -y |

## 利用ansible的service模块实现两台主机的ftp服务的启动。

|  |
| --- |
| $ ansible all -m service -a ‘name=vsftpd state=started’ |

1. 验证启动结果

|  |
| --- |
| $ systemctl status vsftpd |

## 执行结果截图（两台主机都需要）

|  |
| --- |
|  |

## 六、cron模块

用于管理计划任务，如下实例：

机器重启时：

|  |
| --- |
| $ ansible all -m cron -a 'name="a job for reboot" special\_time=reboot job="date>>/tmp/2.txt "' |

每隔一分钟：

|  |
| --- |
| $ ansible all -m cron -a 'name="yum autoupdate" minute=\*/1 hour=\* day=\* month=\* weekday=\* user="root" job="date>>/tmp/1.txt"' |

每个小时1分钟：

|  |
| --- |
| $ ansible all-m cron -a 'name="yum autoupdate" minute=1 hour=\*/1 day=\* month=\* weekday=\* user="root" job="date>>/tmp/1.txt"' |

每5点和2点执行：

|  |
| --- |
| $ ansible all -m cron -a 'backup="True" name="test" minute="0" hour="5,2" job="ls -alh > /dev/null"' |

删除计划任务：

|  |
| --- |
| $ ansilbe all-m cron -a 'cron\_file=ansible\_yum-autoupdate state=absent' |

常用参数：

name #该任务的描述;

backup #对远程主机上的原任务计划内容修改之前做备份;

cron\_file #如果指定该选项，则用该文件替换远程主机上的cron.d目录下的用户的任务计划;

day #日（1-31，，/2,……）;

hour #小时（0-23，，/2，……）;

minute #分钟（0-59，，/2，……）;

month #月（1-12，，/2，……）;

weekday #周（0-7，\*，……）;

job #要执行的任务,依赖于state=present;

special\_time #指定什么时候执行，参数：reboot(重启时),yearly(每年),annually,monthly,weekly,daily,hourly;

state #确认该任务计划是创建还是删除;

user #以哪个用户的身份执行;

## 七、yum模块

使用yum包管理器来管理软件包，实例如下：

# 安装最新版本的apache;

|  |
| --- |
| $ ansible all -m yum -a 'name=httpd state=latest' |

# 移除apache;

|  |
| --- |
| $ ansible all -m yum -a 'name=httpd state=absent' |

# 升级所有的软件包;

|  |
| --- |
| $ ansible all -m yum -a 'name=\* state=latest ' |

# 安装整个Development tools相关的软件包;

|  |
| --- |
| $ ansible all -m yum -a 'name="@Development tools" state=present' |

# 从本地仓库安装nginx;

|  |
| --- |
| $ ansible test -m yum -a 'name=/root/ansible/nginx-1.10.0-1.el7.ngx.x86\_64.rpm state=present' |

# 从一个远程yum仓库安装nginx;

|  |
| --- |
| $ ansible all -m yum -a 'name= http://nginx.org/packages/centos/7/x86\_64/RPMS/nginx-1.10.0-1.el7.ngx.x86\_64.rpm state=present' |

常用参数：

name #要进行操作的软件包的名字，也可以传递一个url或者一个本地的rpm包的路径;

config\_file #yum的配置文件;

disable\_gpg\_check #关闭gpg\_check;

disablerepo #不启用某个源;

enablerepo #启用某个源;

state #用于描述安装包最终状态，present/latest用于安装包，absent用于卸载安装包;

## 八、user模块

用户管理模块，使用实例：

|  |
| --- |
| $ ansible all -m user -a 'name=ansible comment="ansible admin" uid=1040 group=root' |

常用参数：

home #指定用户的家目录，需要与createhome配合使用;

groups #指定用户的属组;

uid #指定用的uid;

password #指定用户的密码;

name #指定用户名;

createhome #是否创建家目录yes|no;

system #是否为系统用户;

remove #当state=absent时，remove=yes则表示连同家目录一起删除，等价于userdel -r;

state #是创建还是删除;

shell #指定用户的shell环境;

注：指定password参数时，不能使用明文密码，因为后面这一串密码会被直接传送到被管理主机的/etc/shadow文件中，所以需要先将密码字符串进行加密处理。然后将得到的字符串放到password中即可。

生成一个密码

|  |
| --- |
| $ echo "123456" | openssl passwd -1 -salt $(< /dev/urandom tr -dc '[:alnum:]' | head -c 32) -stdin |

用上面生成的密码创建用户

|  |
| --- |
| $ ansible all -m user -a 'name=foo password="$1$4P4PlFuE$ur9ObJiT5iHNrb9QnjaIB0"' |

不同的发行版默认使用的加密方式可能会有区别，具体可以查看/etc/login.defs文件确认，centos 6.5及Centos7版本使用的是SHA512加密算法。

## 九、group模块

组管理模块，使用实例：

|  |
| --- |
| $ ansible all -m group -a 'name=ansible state=present' |

常用参数：

gid #指定gid;

name #指定组名称;

state #操作状态，present,absent;

system #是否为系统组;

## 十、filesystem模块

在块设备上创建文件系统，示例如下：

$ ansible test -m filesystem -a 'fstype=ext2 dev=/dev/sdb1 force=yes'

$ ansible test -m filesystem -a 'fstype=ext4 dev=/dev/sdb1 opts="-cc"'

常用参数：

dev #目标块设备;

force #在一个已有文件系统的设备上强制创建;

fstype #文件系统的类型;

opts #传递给mkfs命令的选项;

## 十一、mount模块

配置挂载点

$ ansible test -m mount 'name=/mnt/dvd src=/dev/sr0 fstype=iso9660 opts=ro state=present'

$ ansible test -m mount 'name=/srv/disk src='LABEL=SOME\_LABEL' state=present'

$ ansible test -m mount 'name=/home src='UUID=b3e48f45-f933-4c8e-a700-22a159ec9077' opts=noatime state=present'

$ ansible test -a 'dd if=/dev/zero of=/disk.img bs=4k count=1024'

$ ansible test -a 'losetup /dev/loop0 /disk.img'

$ ansible test -m filesystem -a 'fstype=ext4 force=yes opts=-F dev=/dev/loop0'

$ ansible test -m mount 'name=/mnt src=/dev/loop0 fstype=ext4 state=mounted opts=rw'

常用参数：

fstype #必选项,挂载文件的类型;

name #必选项,挂载点;

opts #传递给mount命令的参数;

src #必选项,要挂载的文件;

state #必选项;

present #只处理fstab中的配置;

absent #删除挂载点;

mounted #自动创建挂载点并挂载之;

umounted #卸载;

## 十二、get\_url模块

该模块主要用于从http、ftp、https服务器上下载文件（类似于wget）

$ ansible all -m get\_url-a 'url=http://ftp5.gwdg.de/pub/linux/archlinux/community/os/x86\_64//vsftpd-3.0.3-4-x86\_64.pkg.tar.xz dest=/etc/ mode=0440'

常用参数：

sha256sum #下载完成后进行sha256 check;

timeout #下载超时时间，默认10s;

url #下载的URL;

url\_password、url\_username #主要用于需要用户名密码进行验证的情况;

use\_proxy #使用代理,代理需事先在环境变更中定义;

## 十三、unarchive模块

用于解压文件的模块，

|  |
| --- |
| $ ansible all -m unarchive -a 'src=/etc/vsftpd-3.0.3-4-x86\_64.pkg.tar.xz dest=/tmp/vsftp/ ' |

$ ansible test -m unarchive -a 'src=/tmp/foo.zip dest=/usr/local/bin copy=no'

$ ansible test -m unarchive -a 'src=https://example.com/example.zip dest=/usr/local/bin copy=no'

常用参数：

copy #在解压文件之前，是否先将文件复制到远程主机，默认为yes。若为no，则要求目标主机上压缩包必须存在;

creates #指定一个文件名，当该文件存在时，则解压指令不执行;

dest #远程主机上的一个路径，即文件解压的路径;

group #解压后的目录或文件的属组;

list\_files #如果为yes，则会列出压缩包里的文件，默认为no，2.0版本新增的选项;

mode #解压后文件的权限;

src #如果copy为yes，则需要指定压缩文件的源路径;

owner #解压后文件或目录的属主;

## 十四、script模块

在指定节点上执行shell/python脚本（注意，该脚本是在ansible控制节点上面的）

$ ansible test -m script -a '/root/src.sh'

## 十五、shell模块

在指定节点上执行shell/python脚本（注意，该脚本是在远程节点）。

|  |
| --- |
| $ ansible test -m shell -a '/bin/bash /root/dest.sh' |

## 十六、command模块

用于执行远程系统命令，此模块为ansible默认执行的模块，也是常用模块之一。

|  |
| --- |
| $ ansible test -m command -a 'uname -n' |

## 十七、raw模块

类似于command模块、区别在于raw模块支持管道传递。

|  |
| --- |
| $ ansible test -m raw -a "tail -n2 /etc/passwd | head -n1" |

## 十八、fetch模块

它用于从远程机器获取文件，并将其本地存储在由主机名组织的文件树中。

– src：远程系统上要获取的文件。 这必须是一个文件，而不是一个目录。 后续版本可能会支持递归提取。

– dest：保存文件的目录。 例如，如果dest目录是/backup，在主机host.example.com上命名为/ etc/profile的src文件将被保存到/backup/host.example.com/etc/profile。

– flat：允许您覆盖将目标文件添加到主机名/ path / to / file的默认行为。

## 十九、pip模块

用于管理Python库依赖项，为了使用pip模块，必须提供参数name或者requirements

– chdir： 执行pip命令前cd进入的目录

– name：要安装的Python库的名称或远程包的URL。

– requirements：一个pip requirements.txt文件的路径，它应该是远程系统的本地文件，如果使用chdir选项，则可以将文件指定为相对路径。

– version：指定的Python库的安装版本。

– extra\_args：额外的参数传递给pip。

– executable：显式可执行文件或可执行文件的路径名，用于为系统中安装的特定版本的Python运行pip。 例如pip-3.3，如果系统中安装了Python 2.7和3.3，并且想要为Python 3.3安装运行pip。 它不能与“virtualenv”参数一起指定（在2.1中添加）。 默认情况下，它将采用适用于python解释器的版本。 pip3在python 3上，pip2或pip在python 2上。

– virtualenv：要安装到的virtualenv目录的可选路径。 它不能与’executable’参数一起指定（在2.1中添加）。 如果virtualenv不存在，则将在安装软件包之前创建它。 可选的virtualenv\_site\_packages，virtualenv\_command和virtualenv\_python选项会影响virtualenv的创建。

– virtualenv\_command：用于创建虚拟环境的命令或路径名。 例如pyvenv，virtualenv，virtualenv2，~/bin /virtualenv，/usr/local/bin/virtualenv。

– virtualenv\_python：用于创建虚拟环境的Python可执行文件。 例如python3.5，python2.7。 未指定时，将使用用于运行ansible模块的Python版本。 当virtualenv\_command使用pyvenv或-m venv模块时，不应使用此参数。

– state：状态（present，absent，latest， forcereinstall），表示是安装还卸载

   present:默认的，表示为安装

   lastest: 安装为最新的版本

   absent：表示删除

   forcereinstall：“forcereinstall”选项仅适用于可ansible 2.1及更高版本。

## 二十、apt模块

这个模块是ubuntu作为远端节点的OS的时候，用的最多的。Apt是啥就不多说了，Ubuntu/Debian的包管理工具。

– deb： 用于安装远程机器上的.deb后缀的软件包（optional）

– install\_recommends：这个参数可以控制远程电脑上是否只是下载软件包，还是下载后安装，默认参数为true,设置为false的时候只下载软件包，不安装

– update\_cache： 当这个参数为yes的时候等于apt-get update（optional）

– name： apt要下载的软件包名字，支持name=git=1.6 这种制定版本的模式

– state：状态（present，absent，latest），表示是安装还卸载

   present:默认的，表示为安装

   lastest: 安装为最新的版本

   absent：表示删除

## 二十一、synchronize模块

– 使用rsync同步文件，将主控方目录推送到指定节点的目录下，其参数如下：

– delete： 删除不存在的文件，delete=yes 使两边的内容一样（即以推送方为主），默认no

– src： 要同步到目的地的源主机上的路径; 路径可以是绝对的或相对的。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

– dest：目的地主机上将与源同步的路径; 路径可以是绝对的或相对的。

– dest\_port：默认目录主机上的端口 ，默认是22，走的ssh协议。

– mode: push或pull，默认push，一般用于从本机向远程主机上传文件，pull 模式用于从远程主机上取文件。

– rsync\_opts：通过传递数组来指定其他rsync选项。

## 二十二、template模块

基于模板方式生成一个文件复制到远程主机（template使用Jinjia2格式作为文件模版，进行文档内变量的替换的模块。它的每次使用都会被ansible标记为”changed”状态。）

– backup： 如果原目标文件存在，则先备份目标文件

– src：在ansible控制器上的Jinja2格式化模板的路径。 这可以是相对或绝对的路径。

– dest：将模板渲染到远程机器上的位置。

force：是否强制覆盖，默认为yes

– owner：目标文件属主

– group：目标文件属组

– mode：目标文件的权限模式，模式可以被指定为符号模式（例如，u + rwx或u = rw，g = r，o = r）。

# 单元七 Playbook介绍

## 1.Playbook介绍

Playbook是Ansible的配置、部署、编排语言。

它可以被描述为一个需要希望远程主机执行命令的方案，或者一组IT程序运行的命令集合。

当执行一些简单的改动时，ansible命令是非常有用的，然而它真的作用在于它的脚本能力。当对一台机器做环境初始化的时候往往需要不止做一件事情，这时使用playbook会更加适合。通过playbook你可以一次在多台机器执行多个指令。通过这种预先设计的配置保持了机器的配置统一，并很简单的执行日常任务。

Playbook还开创了很多特性，它可以允许你传输某个命令的状态到后面的指令，如你可以从一台机器的文件中抓取内容并附为变量，然后在另一台机器中使用，这使得你可以实现一些复杂的部署机制，这是ansible命令无法实现的。

在如右的连接中: [ansible-examples repository](https://github.com/ansible/ansible-examples)，有一些整套的playbooks，它们阐明了上述的这些技巧。建议在浏览器中打开它看看，配合本章节一起看。

## 2.YAML介绍

Ansible使用标准的YAML解析器，使用YAML文件语法即可书写playbook。

YAML是一个可读性高的用来表达资料序列的格式，YAML参考了其他多种语言，包括：XML、C语言、Python、Perl以及电子邮件格式RFC2822等。Clark Evans在2001首次发表了这种语言。

YAML Ain’t Makup Language，即YAML不是XML。不过，在开发这种语言时，YAML的意思是：Yet Another Makrup Language（仍是一种标记语言），**2.1YAML特性：**

YAML的可读性好

YAML和脚本原因的交互性好

YAML有一个一致的信息模型

YAML易于实现

YAML可以基于流来处理

YAML表达能力强，扩展性好。

更多的内容及规范参见www.yaml.org。

### 2.2Playbook组成结构

Inventory

Modules

Ad Hot Commands

Playbooks

Variables #变量元素,可传递给Tasks/Templates使用;

Tasks #任务元素,即调用模块完成任务;

Templates #模板元素,可根据变量动态生成配置文件;

Hadlers #处理器元素,通常指在某事件满足时触发的操作;

Roles #角色元素

### 2.3 Playbook执行

使用Playbook时通过ansible-playbook命令使用，它的参数和ansible命令类似，如参数-k(–ask-pass) 和-K (–ask-sudo) 来询问ssh密码和sudo密码，-u指定用户，这些指令也可以通过规定的单元写在playbook里。

ansible-playbook的简单使用方法：

|  |
| --- |
| $ ansible-playbook /etc/ansible/roles/sites.yml |

也可以并行执行playbook，这里的示例是并行的运行playbook，并行的级别是10个进程：

|  |
| --- |
| $ ansible-playbook /etc/ansible/roles/sites.yml -f 10 |

以下是一个简单的playbook例子，其中仅包含一个 play（play可以有多个）:

|  |
| --- |
| ---  - hosts: all  vars:  http\_port: 80  max\_clients: 200  remote\_user: root  tasks:  - name: ensure apache is at the latest version  yum: name=httpd state=latest  - name: write the apache config file  template: src=/srv/httpd.j2 dest=/etc/httpd.conf  notify:  - restart apache  - name: ensure apache is running  service: name=httpd state=started  handlers:  - name: restart apache  service: name=httpd state=restarted |

### 2.4playbook语言特性。

#### 2.4.1Playbook基础组件

##### 2.4.1.1主机（hosts）和用户（users）

Playbook中的每一个paly的目的都是为了让某个或某些主机以某个指定的用户的身份执行任务，hosts用于指定要执行任务的主机，其可以是一个或多个由冒号分隔主机组，这和前面章节ansible命令提到的的hosts使用一样的语法。remote\_user则用于指定远程主机上的执行任务的用户，如上面示例中：

|  |
| --- |
| - hosts: test  remote\_user: root |

不过，remote\_user也可用于各task中：

|  |
| --- |
| - hosts: test  remote\_user: root  tasks:  - name: test connection  ping:  remote\_user: yourname |

也可以通过指定其通过sudo的方式在远程主机上执行任务，其可用于play全局或某task。此外，甚至可以在sudo时使用sudo\_user指定sudo时切换的用户。

|  |
| --- |
| - hosts: test  remote\_user: root  sudo: yes  sudo\_user: yourname |

如果你需要在使用sudo时指定密码，可在运行ansible-playbook命令时加上选项 –ask-sudo-pass(-K)。如果使用sudo时，playbook疑似被挂起，可能是在sudo prompt处被卡住，这时可执行Control-C杀死卡住的任务，再重新运行一次。

##### 2.4.1.2 Tasks列表

Play的主体部分是task列表，task列表中的各任务按次序逐个在hosts中指定的主机上执行，即在所有主机上完成第一个任务后再开始第二个任务。

在运行playbook 时（从上到下执行），如果一个host执行task失败，这个host将会从整个playbook的rotation中移除，如果发生执行失败的情况，请修正playbook 中的错误，然后重新执行即可。

Task的目的是使用指定的参数执行模块，而在模块参数中可以使用变量，模块执行是幂等的，这意味着多次执行是安全的，因为其结果一致。

对于command module和shell module，重复执行playbook，实际上是重复运行同样的命令。如果执行的命令类似于‘chmod’ 或者‘setsebool’这种命令，这没有任何问题，也可以使用一个叫做‘creates’的flag使得这两个module变得具有”幂等”特性 （不是必要的）。

每一个task必须有一个名称name，这样在运行playbook时，从其输出的任务执行信息中可以很好的辨别出是属于哪一个task的。如果没有定义name，‘action’的值将会用作输出信息中标记特定的task。

定义一个task，以前有一种格式: “action: module options” （可能在一些老的playbooks中还能见到），现在推荐使用更常见的格式:”module: options”，本文档使用的就是这种格式。

|  |
| --- |
| - hosts: test  vars:  http\_port: 80  max\_clients: 200  remote\_user: root  tasks:  - name: ensure apache is at the latest version  yum: name=httpd state=latest |

意思是，针对test组执行一个task使用root用户执行。而这个task任务是“ensure apache is at the latest version”，然后调用yum模块进行httpd的检测，这里的模块参数含义都跟前面在介绍常用模块时是一样的，所以详情可以看【单元六：ansible常用模块介绍】。

在众多模块中，只有command和shell模块仅需要给定一个列表而无需使用”key-value”格式，例如：

|  |
| --- |
| tasks:  - name: disable selinux  command: /sbin/setenforce 0 |

在使用command和shell模块时，我们需要关心返回码信息，如果有一条命令，它的成功执行的返回码不是0（ansible可能会终止运行），我们可以使用如下方式替代，强制返回成功：

|  |
| --- |
| tasks:  - name: run this command and ignore the result  shell: /usr/bin/somecommand || /bin/true |

或者使用ignore\_error来忽略错误信息：

|  |
| --- |
| tasks:  - name: run this command and ignore the result  shell: /usr/bin/somecommand  ignore\_errors: True |

##### 2.4.1.3变量（variables）和模板（template）

在Playbook中可以直接定义变量（当然，前面我们也在inventory中添加变量），如下所示：

|  |
| --- |
| - hosts: test  vars:  - package: httpd |

其中vars是固定格式，而package就是变量名，httpd就是变量值。

但在定义变量名时有一些规范，也就是合法的变量名。在使用变量之前最好先知道什么是合法的变量名，

（1）变量名可以为字母，数字以及下划线。

（2）变量始终应该以字母开头，

“foo\_port”是个合法的变量名。

”foo5”也是，

“foo-port”， “foo port”，“foo.port” 和“12”则不是合法的变量名。

对于定义好的变量可以在task或者template中使用，但一般playbook中定义的变量都是给模板使用的，Ansible使用jijia2模板框架。这个后面会详解。现在只需要知道变量可以在task和template中使用即可。如下在task中调用变量：

|  |
| --- |
| asks:  - name: ensure apache is at the latest version  yum: name={{ package }} state=latest |

在task或template中引用变量都是使用双花括号中间引入变量名即可，如：{{ VAR\_NAME }}。

另外需要说明，在playbook中调用的变量不仅仅是在playbook中自定义的变量，也可以是Ansible中所定义的所有变量。比如说在Ansible中有一个setup模块，就是用来获取客户端所有信息的，如内存、CPU、IP、主机名、磁盘等等信息，这些信息都是key-value格式的。每当我们执行playbook时首先会需要执行setup模块，如下示例：

TASK [setup] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [10.0.60.143]

这意味着什么呢？也就是说当我执行playbook时，其实是可以在task或template中去引用setup模块输出的变量。有哪些变量可用，你可以执行ansible命令看看输出结果：

$ ansible test -m setup -k

10.0.60.143 | SUCCESS => {

"ansible\_facts": {

"ansible\_all\_ipv4\_addresses": [

"10.0.60.143"

],

"ansible\_all\_ipv6\_addresses": [

"fe80::21d:d8ff:feb7:1e2b"

],

.......忽略........

其中变量”ansible\_all\_ipv4\_addresses”和”ansible\_all\_ipv6\_addresses”都可以在task或者template中使用双花括号去引用的。

##### 2.4.1.4 Handlers

Handlers其实挺好理解的，就是在发生改变时执行提前定义好的操作。

上面我们曾提到过，模块具有”幂等”性，所以当远端系统被人改动时，可以重放playbook达到恢复的目的。playbook本身可以识别这种改动，并且有一个基本的event system（事件系统），可以响应这种改动。

（当发生改动时）’notify’actions会在playbook的每一个task结束时被触发，而且即使有多个不同的task通知改动的发生，’notify’ actions只会被触发一次。

这里有一个例子，当一个文件的内容被改动时，重启两个services：

|  |
| --- |
| - name: write the apache config file  copy: src=/srv/httpd.conf dest=/etc/httpd/conf/httpd.conf  notify:  - restart httpd |

“notify”下列出的即是handlers。

Handlers也是一些task的列表，通过名字来引用，它们和一般的task并没有什么区别。Handlers是由通知者进行notify，如果没有被 notify，handlers不会执行。不管有多少个通知者进行了notify，等到play中的所有task执行完成之后，handlers也只会被执行一次。

这里是一个handlers的示例：

|  |
| --- |
| handlers:  - name: restart apache  service: name=apache state=restarted |

下面给一个安装服务然后使用Handlers的例子：

|  |
| --- |
| - hosts: test  remote\_user: root  tasks:  - name: ensure apache is at the latest version  yum: name=httpd state=latest  - name: write the apache config file  copy: src=/srv/httpd.conf dest=/etc/httpd/conf/httpd.conf  - name: ensure apache is running  service: name=httpd state=started |

这个play就做了三个task，第一是检测httpd是否是最新版，如果不是就进行安装；第二就是从服务器copy一份定义好的配置文件到目标主机；第三就是启动httpd服务器。

首选需要在ansible服务器上提供一个/srv/httpd.conf配置文件，不然执行时会报错哦。然后使用ansible-playbook执行。

|  |
| --- |
| $ ansible-playbook apache.yml |

如果有报错看看报错信息，一般会报什么selinux或要安装libselinux-python，在客户端执行，或者指定定义到playbook中：

|  |
| --- |
| $ setenforce 0  $ yum install libselinux-python |

那么此时你更改了配置文件，是不是需要reload httpd或者restart httpd后你的配置文件才能生效。但是上面的这份配置就算你改完配置文件再次执行playbook，配置文件虽然复制过去了，但由于ansible幂等性配置是无法生效的。而Handlers就是为了解决此类问题而生的，我们可以给copy文件的那个task定义一个handlers，一旦有配置文件改动就调用提前定义的handlers进行处理。如下：

|  |
| --- |
| - hosts: test  remote\_user: root  tasks:  - name: ensure apache is at the latest version  yum: name=httpd state=latest  - name: write the apache config file  copy: src=/srv/httpd.conf dest=/etc/httpd/conf/httpd.conf  notify:  - restart httpd  - name: ensure apache is running  service: name=httpd state=started  handlers:  - name: restart httpd  service: name=httpd state=restarted |

这个时候再次修改配置文件，然后执行playbook，当执行到第二个任务后检测到定义了notify，当整个play的task都执行完毕后，就会找handlers，匹配到名称后就会执行对应的模块。

注意：notify定义的restart httpd必须与handlers定义的name名称相同，不然就会报：not found in any of the known handlers

##### 2.4.1.5 Register

经常在playbook中，存储某个命令的结果在变量中，以备日后访问是很有用的。而‘register’就是用来决定把结果存储在哪个变量中，结果参数可以用在模版中，动作条目，或者when语句。像下面这个例子：

|  |
| --- |
| - name: test play  hosts: all  tasks:  - shell: cat /etc/motd  register: motd\_contents  - shell: echo "motd contains the word hi">>1.txt  when: motd\_contents.stdout.find('hi') != -1 |

就像上面展示的那样，这个注册后的参数的内容为字符串’stdout’是可以访问。

这个注册了以后的结果，如果像上面展示的，可以转化为一个list（或者已经是一个list），就可以在任务中的”with\_items”中使用，“stdout\_lines”在对象中已经可以访问了，当然如果你喜欢也可以调用“home\_dirs.stdout.split()” , 也可以用其它字段切割：

|  |
| --- |
| - name: registered variable usage as a with\_items list  hosts: all  tasks:  - name: retrieve the list of home directories  command: ls /home  register: home\_dirs  - name: add home dirs to the backup spooler  file: dest=/mnt/bkspool/{{ item }} src=/home/{{ item }} state=link  with\_items: “{{ home\_dirs.stdout\_lines }}”  # same as with\_items: home\_dirs.stdout.split() |

##### 2.4.1.6提示与技巧

在playbook执行输出信息的底部，可以找到关于托管节点的信息。

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

10.0.60.143 : ok=5 changed=2 unreachable=0 failed=0

其中：

ok：表示执行成功但没有做过任何变动的任务。

changed：表示执行成功但做过变动的任务。

unreachable：表示无法到达的任务。

failed：表示失败的任务。

如果你想看到执行成功的模块的输出信息，使用–verbose即可（否则只有执行失败的才会有输出信息）。

|  |
| --- |
| $ ansible-playbook playbook.yml -verbose |

如果安装了cowsay软件包，ansible playbook的输出已经进行了广泛的升级，可以尝试一下！

在执行一个playbook之前，想看看这个playbook 的执行会影响到哪些 hosts,你可以这样做:

|  |
| --- |
| $ ansible-playbook playbook.yml --list-hosts |

##### 2.4.1.7Playbook其他特性

**条件测试**

如果需要根据变量、facts（setup）或此前任务的执行结果来作为某task执行与否的前提时要用到条件测试。在Playbook中条件测试使用when子句。在task后添加when子句即可使用条件测试：when子句支持jinjia2表达式或语法，例如：

|  |
| --- |
| tasks:  - name: "shutdown Debian"  command: /sbin/shutdown -h now  when: ansible\_os\_family == "Debian" |

或者多条件判断

|  |
| --- |
| tasks:  - name: "shut down CentOS 6 systems"  command: /sbin/shutdown -t now  when:  - ansible\_distribution == "CentOS"  - ansible\_distribution\_major\_version == "6" |

意思就是如果对方的系统是CentOS并且版本是6就进行关机处理，ansible\_os\_family这个变量是从facts中取的。

也可以使用组条件判断：

|  |
| --- |
| tasks:  - name: "shut down CentOS 6 and Debian 7 systems"  command: /sbin/shutdown -t now  when: (ansible\_distribution == "CentOS" and ansible\_distribution\_major\_version == "6") or  (ansible\_distribution == "Debian" and ansible\_distribution\_major\_version == "7") |

另外也可以自定义变量，当值为某某时执行什么动作，如下：

|  |
| --- |
| - hosts: all  vars:  exist: "True"  tasks:  - name: creaet file  command: touch /tmp/test.txt  when: exist | match("True")  - name: delete file  command: rm -rf /tmp/test.txt  when: exist | match("False") |

意思是，当exist变量值为True时就创建文件，当值为False时就删除文件。

**迭代**

当有需要重复性执行的任务时，可以使用迭代机制。其使用格式为将需要迭代的内容定义为item变量引用，并通过with\_items语句指明迭代的元素列表即可。例如：

|  |
| --- |
| tasks:  - name: "Install Packages"  yum: name={{ item }} state=latest  with\_items:  - httpd  - mysql-server  - php |

事实上，with\_items中可以使用元素还可为hashes，例如添加用户和组：

|  |
| --- |
| tasks:  - name: "Add users"  user: name={{ item.name }} state=present groups={{ item.groups }}  with\_items:  - { name:'test1', groups:'wheel'}  - { name:'test2', groups:'root'} |

其中引用变量时前缀item变量是固定的，而item后跟的键名就是在with\_items中定义的字典键名。

迭代功能在做批量处理时非常有用，比如系统初始化时一般要安装很多初始化的包就可以使用迭代了。

关于循环的更多高级功能请看Ansible中文官方：http://ansible.com.cn/docs/playbooks\_loops.html。

**tags**

在一个playbook中，我们一般会定义很多个task，如果我们只想执行其中的某一个task或多个task时就可以使用tags标签功能了，格式如下：

|  |
| --- |
| tasks:  - name: "Copy hosts file"  copy: src=/etc/hosts dest=/etc/hosts  tags:  - hosts |

为复制hosts文件定义了一个tags，tags为hosts。那么在执行此playbook时可通过ansible-playbook命令使用–tags选项能实现仅运行指定的tasks而非所有的，如下：

|  |
| --- |
| $ ansible-playbook playbook.yml --tags="hosts" |

事实上，不光可以为单个或多个task指定同一个tags。playbook还提供了一个特殊的tags为always。作用就是当使用always当tags的task时，无论执行哪一个tags时，定义有always的tags都会执行。

|  |
| --- |
| tasks:  - name: "Copy hosts file"  copy: src=/etc/hosts dest=/etc/hosts  tags:  - always |

##### 2.4.1.8 Playbook总结

当你了解了Playbook的以上基础知识后基本可以使用playbook来编排一些自动化任务了。上面介绍了playbook的Hos、Users、Tasks、Variables、Template等基本组成元素的使用，以及条件测试、迭代、tags等特性。下一章节就介绍playbook中非常重要的一个概念roles。另外对于playbook的template元素，不光可以简单地调用变量，并且支持各种操作符。在使用playbook时，模板也是一个不可或缺的功能。

## 3.对托管节点的要求

通常我们使用 ssh 与托管节点通信，默认使用 sftp.如果 sftp 不可用，可在 ansible.cfg 配置文件中配置成 scp 的方式. 在托管节点上也需要安装 Python 2.4 或以上的版本.如果版本低于 Python 2.5 ,还需要额外安装一个模块:

python-simplejson

注意事项

一、没安装python-simplejson,也可以使用Ansible的”raw”模块和script模块,因此从技术上讲,你可以通过Ansible的”raw”模块安装python-simplejson,之后就可以使用Ansible的所有功能了.

二、如果托管节点上开启了SElinux,你需要安装libselinux-python,这样才可使用Ansible中与copy/file/template相关的函数.你可以通过Ansible的yum模块在需要的托管节点上安装libselinux-python.

三、Python 3 与 Python 2 是稍有不同的语言,大多数Python程序还不能在 Python 3 中正确运行.一些Linux发行版(Gentoo, Arch)没有默认安装 Python 2.X 解释器.在这些系统上,你需要安装一个 Python 2.X 解释器,并在 inventory (详见 Inventory文件) 中设置 ‘ansible\_python\_interpreter’ 变量指向你的 2.X Python.你可以使用 ‘raw’ 模块在托管节点上远程安装Python 2.X.

例如：ansible myhost –sudo -m raw -a “yum install -y python2 python-simplejson” 这条命令可以通过远程方式在托管节点上安装 Python 2.X 和 simplejson 模块.

Red Hat Enterprise Linux, CentOS, Fedora, and Ubuntu 等发行版都默认安装了 2.X 的解释器,包括几乎所有的Unix系统也是如此.

作业要求：

1. 实现批量安装JDK

操作步骤：

1：把压缩包上传到控制主机/opt/ ------------手工操作

2:创建一个目录:/usr/jdk64/ -------------file模块

3：解压jdk的压缩包，解压路径：/usr/jdk64--------------unarchive模块

1. 在/etc/profile配置环境变量JAVA\_HOME

操作步骤：

1：在/etc/profile中新增环境变量配置 ----------------file模块/command模块（echo “export JAVA\_HOME=/usr/jdk64/jdk1.8.0\_11” >>/etc/profile)

export JAVA\_HOME=/usr/jdk64/jdk1.8.0\_11

export PATH=$PATH:$JAVA\_HOME/bin

2：重新加载/etc/profile这个文件，使环境变量生效--------------command模块

1. 验证java -version

操作步骤：

1：执行java -version命令 --------shell模块/raw模块

# 单元八 ansible命令

## 1.ansible:

只执行 单条命令，后面跟主机和选项部分

ansible –h

|  |
| --- |
| [root@localhost roles]# ansible -h  Usage: ansible <host-pattern> [options]  Define and run a single task 'playbook' against a set of hosts  Options:  -a MODULE\_ARGS, --args=MODULE\_ARGS  module arguments  --ask-vault-pass ask for vault password  -B SECONDS, --background=SECONDS  run asynchronously, failing after X seconds  (default=N/A)  -C, --check don't make any changes; instead, try to predict some  of the changes that may occur  -D, --diff when changing (small) files and templates, show the  differences in those files; works great with --check  -e EXTRA\_VARS, --extra-vars=EXTRA\_VARS  set additional variables as key=value or YAML/JSON, if  filename prepend with @  -f FORKS, --forks=FORKS  specify number of parallel processes to use  (default=5)  -h, --help show this help message and exit  -i INVENTORY, --inventory=INVENTORY, --inventory-file=INVENTORY  specify inventory host path or comma separated host  list. --inventory-file is deprecated  -l SUBSET, --limit=SUBSET  further limit selected hosts to an additional pattern  --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)  -M MODULE\_PATH, --module-path=MODULE\_PATH  prepend colon-separated path(s) to module library  (default=[u'/root/.ansible/plugins/modules',  u'/usr/share/ansible/plugins/modules'])  -o, --one-line condense output  --playbook-dir=BASEDIR  Since this tool does not use playbooks, use this as a  subsitute playbook directory.This sets the relative  path for many features including roles/ group\_vars/  etc.  -P POLL\_INTERVAL, --poll=POLL\_INTERVAL  set the poll interval if using -B (default=15)  --syntax-check perform a syntax check on the playbook, but do not  execute it  -t TREE, --tree=TREE log output to this directory  --vault-id=VAULT\_IDS the vault identity to use  --vault-password-file=VAULT\_PASSWORD\_FILES  vault password file  -v, --verbose verbose mode (-vvv for more, -vvvv to enable  connection debugging)  --version show program's version number and exit  Connection Options:  control as whom and how to connect to hosts  -k, --ask-pass ask for connection password  --private-key=PRIVATE\_KEY\_FILE, --key-file=PRIVATE\_KEY\_FILE  use this file to authenticate the connection  -u REMOTE\_USER, --user=REMOTE\_USER  connect as this user (default=None)  -c CONNECTION, --connection=CONNECTION  connection type to use (default=smart)  -T TIMEOUT, --timeout=TIMEOUT  override the connection timeout in seconds  (default=10)  --ssh-common-args=SSH\_COMMON\_ARGS  specify common arguments to pass to sftp/scp/ssh (e.g.  ProxyCommand)  --sftp-extra-args=SFTP\_EXTRA\_ARGS  specify extra arguments to pass to sftp only (e.g. -f,  -l)  --scp-extra-args=SCP\_EXTRA\_ARGS  specify extra arguments to pass to scp only (e.g. -l)  --ssh-extra-args=SSH\_EXTRA\_ARGS  specify extra arguments to pass to ssh only (e.g. -R)  Privilege Escalation Options:  control how and which user you become as on target hosts  -s, --sudo run operations with sudo (nopasswd) (deprecated, use  become)  -U SUDO\_USER, --sudo-user=SUDO\_USER  desired sudo user (default=root) (deprecated, use  become)  -S, --su run operations with su (deprecated, use become)  -R SU\_USER, --su-user=SU\_USER  run operations with su as this user (default=None)  (deprecated, use become)  -b, --become run operations with become (does not imply password  prompting)  --become-method=BECOME\_METHOD  privilege escalation method to use (default=sudo),  valid choices: [ sudo | su | pbrun | pfexec | doas |  dzdo | ksu | runas | pmrun | enable | machinectl ]  --become-user=BECOME\_USER  run operations as this user (default=root)  --ask-sudo-pass ask for sudo password (deprecated, use become)  --ask-su-pass ask for su password (deprecated, use become)  -K, --ask-become-pass  ask for privilege escalation password  Some modules do not make sense in Ad-Hoc (include, meta, etc) |

ansible 127.0.0.1-a 'date' //-a 跟模块的参数，默认使用的是cmmand模块

ansible 127.0.0.1-m shell-a 'date' //-m 指定执行的模块

ansible all-i /etc/ansible/hosts-a 'date' //-i:指定inventory目录

ansibleall -a 'date' -k //all- host文件里的所有组，-k 手动输入密码

ansible all -u root-a 'date' //-u 指定远程主机上的用户名

## 2.ansible-doc

用户查看模块的信息，常用-l 和 -s

ansible-doc -l //列出所有已安装的模块

ansible-doc -l | grep ping

|  |
| --- |
| [root@localhost roles]# ansible-doc -l | grep ping  ios\_ping Tests reachability using ping from Cisco IOS network devices  lambda\_event Creates, updates or deletes AWS Lambda function event mappings.  net\_ping Tests reachability using ping from a network device  netapp\_e\_lun\_mapping NetApp E-Series create, delete, or modify lun mappings  nxos\_igmp\_snooping Manages IGMP snooping global configuration.  nxos\_ping Tests reachability using ping from Nexus switch.  ping Try to connect to host, verify a usable python and return `pong' on success  pingdom Pause/unpause Pingdom alerts  sefcontext Manages SELinux file context mapping definitions  win\_ping A windows version of the classic ping module |

ansible-doc -s ping //查看具体模块的用法

|  |
| --- |
| [root@localhost roles]# ansible-doc -s ping  - name: Try to connect to host, verify a usable python and return `pong' on success  ping:  data: # Data to return for the `ping' return value. If this parameter is set to `crash', the module will cause an exception. |

## 3.ansible-galaxy

指方便从https://galaxy.ansible.com/站点下载第三方扩展模块

ansible-galaxy install xxxx //输入要下载的模块名称

## 4.ansible-lint

是对playbook的语法进行检查的一个工具

ansible-init playbook.yml

## 5.ansible-playbook

通过读取playbook文件后，执行相应的动作。

## 6.ansible-pull

该指令使用需要谈到ansible的另一种模式

pull 模式，这和我们平常经常用的push模式刚好相反，

其适用于以下场景：你有数量巨大的机器需要配置，即使使用非常高的线程还是要花费很多时间；你要在一个没有网络连接的机器上运行Anisble，比如在启动之后安装。

## 7.ansible-vault

ansible-vault主要应用于配置文件中含有敏感信息，又不希望他能被人看到，vault可以帮你加密/解密这个配置文件，属高级用法。

主要对于playbooks里比如涉及到配置密码或其他变量时，可以通过该指令加密，这样我们通过cat看到的会是一个密码串类的文件，编辑的时候需要输入事先设定的密码才能打开。

这种playbook文件在执行时，需要加上 –ask-vault-pass参数，同样需要输入密码后才能正常执行。

# 单元九 项目案例实训

## 9.1 批量安装JDK及配置环境变量