ansible自动化部署

来源：<http://breezey.blog.51cto.com/2400275/1555530>

ansible是新出现的自动化运维工具，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。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。

**Ansible1：简介与基本安装**

Ansible是一个综合的强大的管理工具,他可以对多台主机安装操作系统,并为这些主机安装不同的应用程序,也可以通知指挥这些主机完成不同的任务.查看多台主机的各种信息的状态等,ansible都可以通过模块的方式来完成。

1、Ansible特性

No agents：不需要再被管理节点上安装客户端，只要有sshd即可

No server：在服务端不需要启动任何服务，只需要执行命令就行

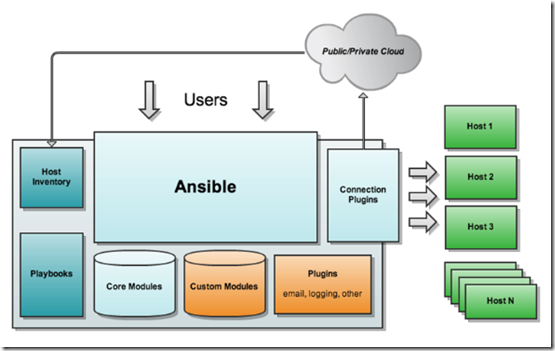
No additional PKI：由于不基于ssl，所以也不基于PKI工作

Modules in any language：基于模块工作，ansible拥有众多的模块

YAML：支持YAML语法

SSH by default：默认使用ssh控制各节点

2、Ansible的基本组件

spacer.gif[](http://s3.51cto.com/wyfs02/M01/7E/32/wKiom1b5C5TCWCv4AAE32f_5jr0656.png)

核心：ansible

核心模块（Core Modules）：这些都是ansible自带的模块

扩展模块（Custom Modules）：如果核心模块不足以完成某种功能，可以添加扩展模块

插件（Plugins）：完成模块功能的补充

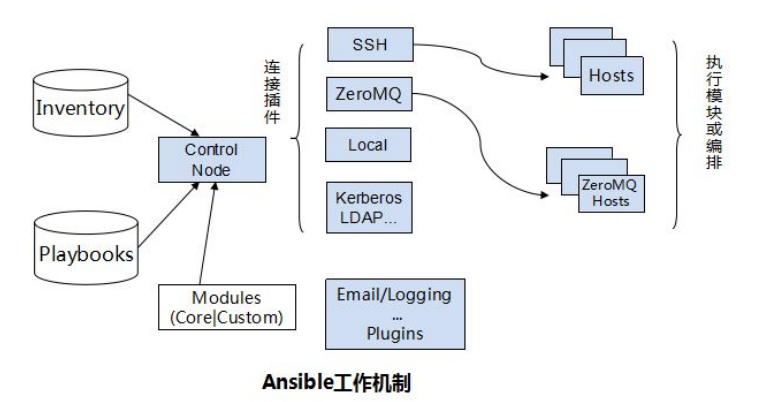
剧本（Playbooks）：把需要完成的多个任务定义在剧本中

连接插件（Connectior Plugins）：ansible基于连接插件连接到各个主机上，虽然ansible是使用ssh连接到各个主机的，但是它还支持其他的连接方法，所以需要有连接插件

主机群（Host Inventory）：ansible在管理多台主机时，可以选择只对其中的一部分执行某些操作

3、Ansible工作机制

Ansible 在管理节点将 Ansible 模块通过 SSH 协议（或者 Kerberos、LDAP）推送到被管理端执 行，执行完之后自动删除，可以使用 SVN 等来管理自定义模块及编排。

spacer.gif[](http://s4.51cto.com/wyfs02/M02/7E/32/wKiom1b5C6yQf5DeAALfOtQONyo494.png)

4、Ansible的安装

Ansible的安装方式有很多种，常用的安装方法是基于yum或者源码，如果是基于yum安装，需要配置epel源，然后直接执行yum -y install ansible即可。源码安装配置如下：

解决依赖关系：

yum -y install python-jinja2 PyYAML python-paramiko python-babel python-crypto

下载ansible：

wget <https://github.com/ansible/ansible/archive/release1.6.1.zip>

解压安装

unzip release1.6.1

cd ansible-release1.6.1

python setup.py build

python setup.py install

mkdir /etc/ansible

cp -r examples/\* /etc/ansible

**用yum安装ansiblel软件（未写）**

**Ansible 安装与简单使用**

转自：<http://blog.chinaunix.net/xmlrpc.php?r=blog/index&uid=29253247&sid=191677&pattern=list&page=1>

**一、安装**

1. $ pip install ansible
2. #for debian
3. $ sudo apt-add-repository ppa:rquillo/ansible
4. $ sudo apt-get update
5. $ sudo apt-get install ansible
6. #for centos
7. $ sudo yum install ansible

**二、环境说明**

1. 控制主机为centos7.3

2. 被控端为 2台 centos7.3 虚拟机 和 一台 centos 6.5虚拟机

3. 通过ssh 密钥认证

4、设置主机组配置文件/etc/ansible/hosts，添加如下内容：

[test]

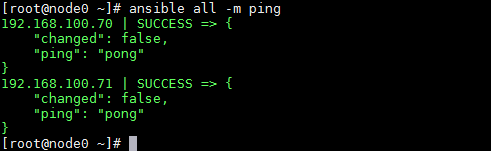
192.168.100.70

192.168.100.71

192.168.100.72

**三、初试牛刀**

1. ansible all -m ping #内建的 ping 模块

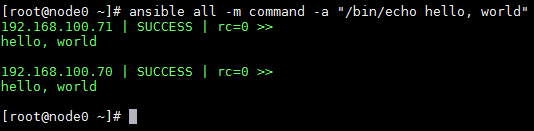


1、ansible 192.168.100.71 -a "/bin/echo hello, world" #执行命令

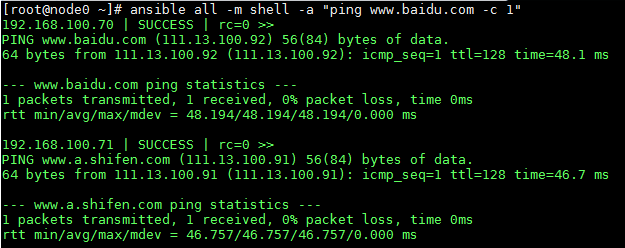
2、ansible all -m command -a "/bin/echo hello, world"

3、#以上两条命令结果是一样的，因为默认调用 command 模块，所以可以省略， all 表示所有节点， -a 后面是参数





1. ansible all -m shell -a "ping www.baidu.com -c 1" # shell 模块



**Ansible2：主机清单**

Ansible 通过读取默认的主机清单配置/etc/ansible/hosts,可以同时连接到多个远程主机上执行任务, 默认路径可以通过修改 ansible.cfg 的 hostfile 参数指定路径。

**一、Hosts and Groups（主机与组）**

对于/etc/ansible/hosts最简单的定义格式像下面：

**1、简单的主机和组**

mail.yanruogu.com

[webservers]

web1.yanruogu.com

web2.yanruogu.com

[dbservers]

db1.yanruogu.com

db2.yanruogu.com

a、中括号中的名字代表组名，可以根据自己的需求将庞大的主机分成具有标识的组，如上面分了两个组webservers和dbservers组；

b、主机(hosts)部分可以使用域名、主机名、IP地址表示；当然使用前两者时，也需要主机能反解析到相应的IP地址，一般此类配置中多使用IP地址；

**2、端口与别名**

如果某些主机的SSH运行在自定义的端口上，ansible使用Paramiko进行ssh连接时，不会使用你SSH配置文件中列出的端口，但是如果修改ansible使用openssh进行ssh连接时将会使用：

192.168.1.1:3091

假如你想要为某些静态IP设置一些别名，可以这样做：

web1 ansible\_ssh\_port = 3333 ansible\_ssh\_host = 192.168.1.2

上面的 web1别名就指代了IP为192.168.1.2，ssh连接端口为3333的主机。

**3、指定主机范围**

[webservers]

www[01:50].yanruogu.com

[databases]

db-[a:f].yanruogu.com

上面指定了从web1到web50，webservers组共计50台主机；databases组有db-a到db-f共6台主机。

**4、使用主机变量**

以下是Hosts部分中经常用到的变量部分：

ansible\_ssh\_host #用于指定被管理的主机的真实IP

ansible\_ssh\_port #用于指定连接到被管理主机的ssh端口号，默认是22

ansible\_ssh\_user #ssh连接时默认使用的用户名

ansible\_ssh\_pass #ssh连接时的密码

ansible\_sudo\_pass #使用sudo连接用户时的密码

ansible\_sudo\_exec #如果sudo命令不在默认路径，需要指定sudo命令路径

ansible\_ssh\_private\_key\_file #秘钥文件路径，秘钥文件如果不想使用ssh-agent管理时可以使用此选项

ansible\_shell\_type #目标系统的shell的类型，默认sh

ansible\_connection #SSH 连接的类型： local , ssh , paramiko，在 ansible 1.2 之前默认是 paramiko ，后来智能选择，优先使用基于 ControlPersist 的 ssh （支持的前提）

ansible\_python\_interpreter #用来指定python解释器的路径，默认为/usr/bin/python 同样可以指定ruby 、perl 的路径

ansible\_\*\_interpreter #其他解释器路径，用法与ansible\_python\_interpreter类似，这里"\*"可以是ruby或才perl等其他语言

示例如下：

[test]

192.168.1.1 ansible\_ssh\_user=root ansible\_ssh\_pass='P@ssw0rd'

192.168.1.2 ansible\_ssh\_user=breeze ansible\_ssh\_pass='123456'

192.168.1.3 ansible\_ssh\_user=bernie ansible\_ssh\_port=3055 ansible\_ssh\_pass='456789'

上面的示例中指定了三台主机，三台主机的用密码分别是P@ssw0rd、123456、45789，指定的ssh连接的用户名分别为root、breeze、bernie，ssh 端口分别为22、22、3055 ，这样在ansible命令执行的时候就不用再指令用户和密码等了。

**5、组内变量**

变量也可以通过组名，应用到组内的所有成员：

[test]

host1

host2

[test:vars]

ntp\_server=192.168.1.10

proxy=192.168.1.20

上面test组中包含两台主机，通过对test组指定vars变更，相应的host1和host2相当于相应的指定了ntp\_server和proxy变量参数值 。

**6、组的包含与组内变量**

[wuhan]

web1

web2

[suizhou]

web4

web3

[hubei:children]

wuhan

suizhou

[hubei:vars]

ntp\_server=192.168.1.10

zabbix\_server=192.168.1.10

[china:children]

hubei

hunan

上面的示例中，指定了武汉组有web1、web2；随州组有web3、web4主机；又指定了一个湖北组，同时包含武汉和随州；同时为该组内的所有主机指定了2个vars变量。设定了一个组中国组，包含湖北、湖南。

注：vars变量在ansible ad-hoc部分中基本用不到，主要用在ansible-playbook中。

**二、Patterns（主机与组正则匹配部分）**

把Patterns 直接理解为正则实际是不完全准确的，正常的理解为patterns意味着在ansible中管理哪些主机，也可以理解为，要与哪台主机进行通信。在探讨这个问题之前我们先看下ansible的用法：

ansible <pattern\_goes\_here> -m <module\_name> -a <arguments>

直接上一个示例：

ansible webservers -m service -a "name=httpd state=restarted"

这里是对webservers 组或主机重启httpd服务 ，其中webservers 就是Pattern部分。而之所以上面说Pattern（模式）可以理解为正则，主要针对下面经常用到的用法而言的。

**1、表示所有的主机可以使用all 或 \***

**2、通配符与逻辑或**

利用通配符还可以指定一组具有规则特征的主机或主机名，冒号表示or－－－逻辑或

web1.yanruogu.com

web1.yanruogu.com:web2.yanruogu.com

192.168.1.1

192.168.1.\*

当然，这里的\*通配符也可以用在前面，如：

\*.yanruogu.com

\*.com

webservers1[0] #表示匹配 webservers1 组的第 1 个主机 webservers1[0:25] #表示匹配 webservers1 组的第 1 个到第 25 个主机（官网文档是":"表示范围，测试发现应该使用"-",注意不要和匹配多个主机组混淆）

上面的用法，在多个组之间同样适用 ，如：

webservers

webservers:dbservers #表示两个组中所有的主机

**3、逻辑非与逻辑and**

非的表达式，如，目标主机必须在组webservers但不在phoenix组中

webserver:!phoenix

交集的表达式，如，目标主机必须即在组webservers中又在组staging中

webservers:&staging

一个更复杂的示例：

webserver:dbservers:&staging:!phoenix

上面这个复杂的表达式最后表示的目标主机必须满足：在webservers或者dbservers组中，必须还存在于staging组中，但是不在phoenix组中 。

**4、混合高级用法**

\*.yanruogu.com:\*.org

还可以在开头的地方使用”~”，用来表示这是一个正则表达式:

~(web|db).\*\.yanruogu\.com

给两个ansible-playbook中具体可能用的用法：

a、在ansible-palybook命令中，你也可以使用变量来组成这样的表达式，但是你必须使用“-e”的选项来指定这个表达式（通常我们不这样用）：

ansible-palybook -e webservers:!{{excluded}}:&{{required}}

b、在ansible和ansible-playbook中，还可以通过一个参数”--limit”来明确指定排除某些主机或组：

ansible-playbook site.yml --limit datacenter2

c、从Ansible1.2开始，如果想排除一个文件中的主机可以使用"@"：

ansible-playbook site.yml --limit @retry\_hosts.txt

**Ansible3：ansible.cfg配置说明**

Ansible默认安装好后有一个配置文件/etc/ansible/ansible.cfg，该配置文件中定义了ansible的主机的默认配置部分，如默认是否需要输入密码、是否开启sudo认证、action\_plugins插件的位置、hosts主机组的位置、是否开启log功能、默认端口、key文件位置等等。

具体如下：

[defaults]

# some basic default values...

hostfile = /etc/ansible/hosts \\指定默认hosts配置的位置

# library\_path = /usr/share/my\_modules/

remote\_tmp = $HOME/.ansible/tmp

pattern = \*

forks = 5

poll\_interval = 15

sudo\_user = root \\远程sudo用户

#ask\_sudo\_pass = True \\每次执行ansible命令是否询问ssh密码

#ask\_pass = True \\每次执行ansible命令时是否询问sudo密码

transport = smart

remote\_port = 22

module\_lang = C

gathering = implicit

host\_key\_checking = False \\关闭第一次使用ansible连接客户端是输入命令提示

log\_path = /var/log/ansible.log \\需要时可以自行添加。chown -R root:root ansible.log

system\_warnings = False \\关闭运行ansible时系统的提示信息，一般为提示升级

# set plugin path directories here, separate with colons

action\_plugins = /usr/share/ansible\_plugins/action\_plugins

callback\_plugins = /usr/share/ansible\_plugins/callback\_plugins

connection\_plugins = /usr/share/ansible\_plugins/connection\_plugins

lookup\_plugins = /usr/share/ansible\_plugins/lookup\_plugins

vars\_plugins = /usr/share/ansible\_plugins/vars\_plugins

filter\_plugins = /usr/share/ansible\_plugins/filter\_plugins

fact\_caching = memory

[accelerate]

accelerate\_port = 5099

accelerate\_timeout = 30

accelerate\_connect\_timeout = 5.0

# The daemon timeout is measured in minutes. This time is measured

# from the last activity to the accelerate daemon.

accelerate\_daemon\_timeout = 30

如果在对之前未连接的主机进行连结时报错如下：

ansible test -a 'uptime'

192.168.1.1| FAILED =>**Using** a SSH password instead of a key is not possible because **HostKey** checking is enabled and sshpass does not support this.**Please** add this host's fingerprint to your known\_hosts file to manage this host.

192.168.1.2 | 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.

是由于在本机的~/.ssh/known\_hosts文件中并有fingerprint key串，ssh第一次连接的时候一般会提示输入yes 进行确认为将key字符串加入到 ~/.ssh/known\_hosts 文件中。

**方法1：**

在进行ssh连接时，可以使用-o参数将StrictHostKeyChecking设置为no，使用ssh连接时避免首次连接时让输入yes/no部分的提示。通过查看ansible.cfg配置文件，发现如下行：

[ssh\_connection]

# ssh arguments to use

# Leaving off ControlPersist will result in poor performance, so use

# paramiko on older platforms rather than removing it

#ssh\_args = -o ControlMaster=auto -o ControlPersist=60s

可以启用ssh\_args 部分，使用下面的配置，避免上面出现的错误：

ssh\_args = -o ControlMaster=auto -o ControlPersist=60s -o StrictHostKeyChecking＝no

**方法2：**

在ansible.cfg配置文件中，也会找到如下配置：

# uncomment this to disable SSH key host checking

host\_key\_checking = False

默认host\_key\_checking部分是注释的，通过找开该行的注释，同样也可以实现跳过ssh 首次连接提示验证部分。但在实际测试中，似乎并没有效果，建议使用方法1.

**其他部分**

默认ansible 执行的时候，并不会输出日志到文件，不过在ansible.cfg 配置文件中有如下行：

log\_path = /var/log/ansible.log

默认log\_path这行是注释的，打开该行的注释，所有的命令执行后，都会将日志输出到/var/log/ansible.log文件。

**Ansible4：Ad-hoc与命令执行模块**

Ad-Hoc 是指ansible下临时执行的一条命令，并且不需要保存的命令，对于复杂的命令会使用playbook。Ad-hoc的执行依赖于模块，ansible官方提供了大量的模块。 如：command、raw、shell、file、cron等，具体可以通过ansible-doc -l 进行查看 。可以使用ansible-doc -s module来查看某个模块的参数，也可以使用ansible-doc help module来查看该模块更详细的信息。

**一、Ad-hoc**

**1、命令说明**

一个ad-hoc命令的执行，需要按以下格式进行执行：

ansible 主机或组-m 模块名-a '模块参数' ansible参数

* 主机和组，是在/etc/ansible/hosts 里进行指定的部分，当然动态Inventory 使用的是脚本从外部应用里获取的主机；
* 模块名，可以通过ansible-doc -l 查看目前安装的模块，默认不指定时，使用的是command模块，具体可以查看/etc/ansible/ansible.cfg 的“#module\_name = command ” 部分，默认模块可以在该配置文件中进行修改；
* 模块参数，可以通过 “ansible-doc -s 模块名” 查看具体的用法及后面的参数；
* ansible参数，可以通过ansible命令的帮助信息里查看到，这里有很多参数可以供选择，如是否需要输入密码、是否sudo等。

**2、后台执行**

当命令执行时间比较长时，也可以放到后台执行，使用-B、-P参数，如下：

ansible all -B 3600-a "/usr/bin/long\_running\_operation --do-stuff" #后台执行命令3600s，-B 表示后台执行的时间

ansible all -m async\_status -a "jid=123456789" #检查任务的状态

ansible all -B 1800-P 60-a "/usr/bin/long\_running\_operation --do-stuff" #后台执行命令最大时间是1800s即30分钟，-P 每60s检查下状态，默认15s

**二、命令执行模块**

命令执行模块包含如下 四个模块：

* [command模块](http://docs.ansible.com/command_module.html)：该模块通过-a跟上要执行的命令可以直接执行，不过命令里如果有带有如下字符部分则执行不成功 “ "<", ">", "|", "&" ；
* [shell 模块](http://docs.ansible.com/shell_module.html)：用法基本和command一样，不过其是通过/bin/sh进行执行，所以shell 模块可以执行任何命令，就像在本机执行一样；
* [raw模块](http://docs.ansible.com/raw_module.html)：用法和shell 模块一样 ，其也可以执行任意命令，就像在本机执行一样；
* [script模块](http://docs.ansible.com/script_module.html)：其是将管理端的shell 在被管理主机上执行，其原理是先将shell 复制到远程主机，再在远程主机上执行，原理类似于raw模块。

注：raw模块和comand、shell 模块不同的是其没有chdir、creates、removes参数，chdir参数的作用就是先切到chdir指定的目录后，再执行后面的命令，这在后面很多模块里都会有该参数 。

command模块包含如下选项：

* creates：一个文件名，当该文件存在，则该命令不执行
* free\_form：要执行的linux指令
* chdir：在执行指令之前，先切换到该指定的目录
* removes：一个文件名，当该文件不存在，则该选项不执行
* executable：切换shell来执行指令，该执行路径必须是一个绝对路径

**使用chdir的示例：**

ansible 192.168.1.1 -m command -a 'chdir=/tmp/test.txt touch test.file'

ansible 192.168.1.1 -m shell -a 'chdir=/tmp/test.txt touch test2.file'

ansible 192.168.1.1 -m raw -a 'chdir=/tmp/text.txt touch test3.file'

三个命令都会返回执行成功的状态。不过实际上只有前两个文件会被创建成功。使用raw模块的执行的结果文件事实上也被正常创建了，不过不是在chdir指定的目录，而是在当前执行用户的家目录。

**creates与removes示例：**

ansible 192.168.1.1 -a 'creates=/tmp/server.txt uptime' #当/tmp/server.txt文件存在时，则不执行uptime指令

ansible 192.168.1.1 -a 'removes=/tmp/server.txt uptime' #当/tmp/server.txt文件不存在时，则不执行uptime指令

**script模块示例：**

要执行的脚本文件script.sh内容如下：

#/bin/bash

ifconfig

df -hT

执行ansible指令：ansible 10.212.52.252 -m script -a 'script.sh' |egrep '>>|stdout'

**Ansible5：常用模块(简介)**

Ansible通过模块的方式来完成一些远程的管理工作。可以通过ansible-doc -l查看所有模块，可以使用ansible-doc -s module来查看某个模块的参数，也可以使用ansible-doc help module来查看该模块更详细的信息。下面列出一些常用的模块：

1. setup

可以用来查看远程主机的一些基本信息：

ansible -i /etc/ansible/hosts test -m setup

2.ping

可以用来测试远程主机的运行状态：

ansible test -m ping

3.file

设置文件的属性

file模块包含如下选项：

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

示例：

ansible test -m file -a "src=/etc/fstab dest=/tmp/fstab state=link"

ansible test -m file -a "path=/tmp/fstab state=absent"

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

4.copy

复制文件到远程主机

copy模块包含如下选项：

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

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

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 ：The validation command to run before copying into place. The path to the file to validate is passed in via '%s' which must be present as in the visudo example below.

示例：

ansible test -m copy -a "src=/srv/myfiles/foo.conf dest=/etc/foo.conf owner=foo group=foo mode=0644"

ansible test -m copy -a "src=/mine/ntp.conf dest=/etc/ntp.conf owner=root group=root mode=644 backup=yes"

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

5.command

在远程主机上执行命令

command模块包含如下选项：

creates：一个文件名，当该文件存在，则该命令不执行

free\_form：要执行的linux指令

chdir：在执行指令之前，先切换到该指定的目录

removes：一个文件名，当该文件不存在，则该选项不执行

executable：切换shell来执行指令，该执行路径必须是一个绝对路径

示例：

ansible test -a "/sbin/reboot"

6.shell

切换到某个shell执行指定的指令，参数与command相同。

示例：

ansible test -m shell -a "somescript.sh >> somelog.txt"

7.service

用于管理服务

该模块包含如下选项：

arguments：给命令行提供一些选项

enabled：是否开机启动 yes|no

name：必选项，服务名称

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

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

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）

示例：

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

ansible test -m service -a "name=foo pattern=/usr/bin/foo state=started"

ansible test -m service -a "name=network state=restarted args=eth0"

8.cron

用于管理计划任务

包含如下选项：

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

name：该任务的描述

special\_time：指定什么时候执行，参数：reboot,yearly,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

示例：

ansible test -m cron -a 'name="check dirs" hour="5,2" job="ls -alh > /dev/null"'

ansible test -m cron -a 'name="a job for reboot" special\_time=reboot job="/some/job.sh"'

ansible test -m cron -a 'name="yum autoupdate" weekday="2" minute=0 hour=12 user="root" job="YUMINTERACTIVE=0 /usr/sbin/yum-autoupdate" cron\_file=ansible\_yum-autoupdate'

ansilbe test -m cron -a 'cron\_file=ansible\_yum-autoupdate state=absent'

9.filesystem

在块设备上创建文件系统

选项：

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

10.yum

使用yum包管理器来管理软件包

选项：

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

list

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

state：状态（present，absent，latest）

示例：

ansible test -m yum -a 'name=httpd state=latest'

ansible test -m yum -a 'name="@Development tools" state=present'

ansible test -m yum -a 'name=http://nginx.org/packages/centos/6/noarch/RPMS/nginx-release-centos-6-0.el6.ngx.noarch.rpm state=present'

11.user

管理用户

home:

groups:

uid

password:

name:

createhome:

system:

remove:

state:

shell:

需要特别说明的是，password后面指定的密码不能是明文，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，而登陆的时候输入的密码会被hash加密以后再去与/etc/shadow中存放的密码去做对比，会出现不一致的现象。所以需要先将密码字符串进行加密处理：openssl passwd -salt -1 "123456"，然后将得到的字符串放到password中即可。

12.group

管理组

13.synchronize

使用rsync同步文件

archive

checksum

delete

dest

src

dest\_port

existing\_only: skip createing new files on receiver

links

owner

mode:(push, pull)

recursive

rsync\_path

times:Preserve modification times

示例：

src=some/relative/path dest=/some/absolute/path rsync\_path="sudo rsync"

src=some/relative/path dest=/some/absolute/path archive=no links=yes

src=some/relative/path dest=/some/absolute/path checksum=yes times=no

src=/tmp/helloworld dest=/var/www/helloword rsync\_opts=--no-motd,--exclude=.git mode=pull

14.mount

配置挂载点

选项：

dump

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

passno

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

示例：

name=/mnt/dvd src=/dev/sr0 fstype=iso9660 opts=ro state=present

name=/srv/disk src='LABEL=SOME\_LABEL' state=present

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 'fstype=ext4 force=yes opts=-F dev=/dev/loop0'

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

15.raw

类似command，但可以传递管道

**Ansible5：常用模块(详解)**

根据zs官方的分类，将模块按功能分类为：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 ，具体可以参看[官方页面](http://docs.ansible.com/list_of_all_modules.html)。

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。

**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

**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：

ansible 10.212.52.252 -m setup -a 'filter=ansible\_\*\_mb' //查看主机内存信息

ansible 10.212.52.252 -m setup -a 'filter=ansible\_eth[0-2]' //查看地接口为eth0-2的网卡信息

ansible all -m setup --tree /tmp/facts //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）

**三、file模块**

file模块主要用于远程主机上的文件操作，file模块包含如下选项：

* 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

使用示例：

ansible test -m file -a "src=/etc/fstab dest=/tmp/fstab state=link"

ansible test -m file -a "path=/tmp/fstab state=absent"

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

**四、copy模块**

复制文件到远程主机，copy模块包含如下选项：

* 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
* 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
* 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 ：The validation command to run before copying into place. The path to the file to validate is passed in via '%s' which must be present as in the visudo example below.

示例如下：

ansible test -m copy -a "src=/srv/myfiles/foo.conf dest=/etc/foo.conf owner=foo group=foo mode=0644"

ansible test -m copy -a "src=/mine/ntp.conf dest=/etc/ntp.conf owner=root group=root mode=644 backup=yes"

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

**五、service模块**

用于管理服务

该模块包含如下选项：

* arguments：给命令行提供一些选项
* enabled：是否开机启动 yes|no
* name：必选项，服务名称
* 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
* sleep：如果执行了restarted，在则stop和start之间沉睡几秒钟
* 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）

使用示例：

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

asnible test -m service -a "name=foo pattern=/usr/bin/foo state=started"

ansible test -m service -a "name=network state=restarted args=eth0"

**六、cron模块**

用于管理计划任务包含如下选项：

* 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
* name：该任务的描述
* special\_time：指定什么时候执行，参数：reboot,yearly,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

示例：

ansible test -m cron -a 'name="a job for reboot" special\_time=reboot job="/some/job.sh"'

ansible test -m cron -a 'name="yum autoupdate" weekday="2" minute=0 hour=12 user="root

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

ansilbe test -m cron -a 'cron\_file=ansible\_yum-autoupdate state=absent'

**七、yum模块**

使用yum包管理器来管理软件包，其选项有：

* 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
* 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
* state：状态（present，absent，latest）

示例如下：

ansible test -m yum -a 'name=httpd state=latest'

ansible test -m yum -a 'name="@Development tools" state=present'

ansible test -m yum -a 'name=http://nginx.org/packages/centos/6/noarch/RPMS/nginx-release-centos-6-0.el6.ngx.noarch.rpm state=present'

**八、user模块与group模块**

user模块是请求的是useradd, userdel, usermod三个指令，goup模块请求的是groupadd, groupdel, groupmod 三个指令。

**1、user模块**

* 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

使用示例：

user: name=johnd comment="John Doe" uid=1040 group=admin

user: name=james shell=/bin/bash groups=admins,developers append=yes user: name=johnd state=absent remove=yes

user: name=james18 shell=/bin/zsh groups=developers expires=1422403387

user: name=test generate\_ssh\_key=yes ssh\_key\_bits=2048 ssh\_key\_file=.ssh/id\_rsa #生成密钥时，只会生成公钥文件和私钥文件，和直接使用ssh-keygen指令效果相同，不会生成authorized\_keys文件。

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。

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

$1$4P4PlFuE$ur9ObJiT5iHNrb9QnjaIB0

#使用上面的密码创建用户

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版本使用的是SHA512加密算法。

**2、group示例**

ansible all -m group -a 'name=somegroup state=present'

**九、synchronize模块**

使用rsync同步文件，其参数如下：

* archive: 归档，相当于同时开启recursive(递归)、links、perms、times、owner、group、-D选项都为yes ，默认该项为开启
* checksum: 跳过检测sum值，默认关闭
* compress:是否开启压缩
* copy\_links：复制链接文件，默认为no ，注意后面还有一个links参数
* delete: 删除不存在的文件，默认no
* dest：目录路径
* dest\_port：默认目录主机上的端口 ，默认是22，走的ssh协议
* dirs：传速目录不进行递归，默认为no，即进行目录递归
* rsync\_opts：rsync参数部分
* set\_remote\_user：主要用于/etc/ansible/hosts中定义或默认使用的用户与rsync使用的用户不同的情况
* mode: push或pull 模块，push模的话，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

使用示例：

src=some/relative/path dest=/some/absolute/path rsync\_path="sudo rsync"

src=some/relative/path dest=/some/absolute/path archive=no links=yes

src=some/relative/path dest=/some/absolute/path checksum=yes times=no

src=/tmp/helloworld dest=/var/www/helloword rsync\_opts=--no-motd,--exclude=.git mode=pull

**十、filesystem模块**

在块设备上创建文件系统

选项：

* 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

示例：

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

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

**十一、mount模块**

配置挂载点

选项：

* dump
* 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

示例：

name=/mnt/dvd src=/dev/sr0 fstype=iso9660 opts=ro state=present

name=/srv/disk src='LABEL=SOME\_LABEL' state=present

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 'fstype=ext4 force=yes opts=-F dev=/dev/loop0'

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

**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），主要有如下选项：

* 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：是事使用代理，代理需事先在环境变更中定义

示例：

get\_url: url=http://example.com/path/file.conf dest=/etc/foo.conf mode=0440

get\_url: url=http://example.com/path/file.conf dest=/etc/foo.conf sha256sum=b5bb9d8014a0f9b1d61e21e796d78dccdf1352f23cd32812f4850b878ae4944c

**十三、unarchive模块**

用于解压文件，模块包含如下选项：

* 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
* grop：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：解决后文件的权限
* 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

示例如下：

- unarchive: src=foo.tgz dest=/var/lib/foo

- unarchive: src=/tmp/foo.zip dest=/usr/local/bin copy=no

- unarchive: src=https://example.com/example.zip dest=/usr/local/bin copy=no

**Ansible6：Playbook简单使用**

ansbile-playbook是一系列ansible命令的集合，利用yaml 语言编写。playbook命令根据自上而下的顺序依次执行。同时，playbook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。

playbook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 。

ansible-playbook的简单使用方法: ansible-playbook example-play.yml 。

**一、一个简单的示例**

下面是一个简单的ansible-playbook示例，可以了解其构成:

# cat user.yml

- name: create user

hosts: all

user: root

gather\_facts: false

vars:

user:"test"

tasks:

- name: create user

user: name="{{ user }}"

上面的playbook 实现的功能是新增一个用户：

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 ；

hosts参数指定了对哪些主机进行参作；

user参数指定了使用什么用户登录远程主机操作；

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，这里指定一个user变量，其值为test ，需要注意的是，变量值一定要用引号引住；

task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。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。

同样，如果想实现把这个新增的用户删除，只需将该playbook文件的最后一行替换为如下行再执行相应的playbook即可：

user: name="{{ user }}" state=absent remove=yes

**二、通过Playbook安装apache示例**

通过ansible-playbook实现对多台主机同时同时安装apache。需要注意的是，多台被管理主机的操作系统可能不相同，而导致apache包名不同，假设同时存在CentOS和Debian两种操作系统，具体playbook内容如下：

# cat install\_apache.yml

- hosts: all

remote\_user: root

gather\_facts:**True**

tasks:

- name: install apache on CentOS

yum: name=httpd state=present

when: ansible\_os\_family =="CentOS"

- name: install apache on Debian

yum: name=apache2 state=present

when: ansible\_os\_family =="Debian"

上面使用了when语句，同时也开启了gather\_facts setup模块，这里的ansible\_os\_family变量和就是直接使用的setup模块获取的信息。如果有大量主机，就在运行的时候加上-f然后选择一个合适的并发主机数量即可。

**三、playbook的构成**

playbook是由一个或多个"play"组成的列表。play的主要功能在于将事先归并为一组的主机装扮成事先通过ansible中的task定义好的角色。从根本上来讲所谓task无非是调用ansible的一个module。将多个play组织在一个playbook中即可以让它们联同起来按事先编排的机制同唱一台大戏。其主要有以下四部分构成:

playbooks组成：

Target 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

而其对应的目录层为五个，如下：

一般所需的目录层有：(视情况可变化)

vars 变量层

tasks 任务层

handlers 触发条件

files 文件

template 模板

**1、Hosts和Users**

playbook中的每一个play的目的都是为了让某个或某些主机以某个指定的用户身份执行任务。

hosts：用于指定要执行指定任务的主机其可以是一个或多个由冒号分隔主机组。

remote\_user ：用于指定远程主机上的执行任务的用户。不过remote\_user也可用于各task中。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全局或某任务。此外甚至可以在sudo时使用sudo\_user指定sudo时切换的用户。

user：于remote\_user相同

sudo：如果设置为yes，执行该任务组的用户在执行任务的时候，获取root权限

sudo\_user：如果设置user为breeze，sudo为yes，sudo\_user为bernie时，则breeze用户在执行任务时会获得bernie用户的权限

connection：通过什么方式连接到远程主机，默认为ssh

gather\_facts：除非明确说明不需要在远程主机上执行setup模块，否则默认自动执行。如果确实不需要setup模块传递过来的变量，则可以将该选项设置为False

示例：

- hosts: webnodes

tasks:

- name: test ping connection:

remote\_user: test

sudo: yes

**2、任务列表和action**

play的主体部分是任务列表。

任务列表中的各任务按次序逐个在hosts中指定的所有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。在自上而下运行某playbook时如果中途发生错误，所有已执行任务都将回滚因此在更正playbook后重新执行一次即可。

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，因为其结果均一致。每个task都应该有其name用于 playbook的执行结果输出，建议其内容尽可能清晰地描述任务执行步骤。如果未提供name则action的结果将用于输出。

定义task的可以使用"action: module options"或"module: options"的格式，推荐使用后者以实现向后兼容。如果action一行的内容过多也可使用在行首使用几个空白字符进行换行。

tasks:

- name: make sure apache is running

service: name=httpd state=running

#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

#如果命令或脚本的退出码不为零可以使用如下方式替代

tasks:

- name: run this command and ignore the result

shell: /usr/bin/somecommand || /bin/true

#使用ignore\_errors来忽略错误信息

tasks:

- name: run this command and ignore the result

shell: /usr/bin/somecommand

ignore\_errors: True

**3、handlers**

用于当关注的资源发生变化时采取一定的操作。

"notify"这个action可用于在每个play的最后被触发，这样可以避免多次有改变发生时每次都执行指定的操作，取而代之仅在所有的变化发生完成后一次性地执行指定操作。

在notify中列出的操作称为handler也即notify中调用handler中定义的操作。

注意：在notify中定义内容一定要和tasks中定义的 - name 内容一样，这样才能达到触发的效果，否则会不生效。

- name: template configuration file

template: src=template.j2 dest=/etc/foo.conf

notify:

- restart memcached

- restart apache

#handler是task列表这些task与前述的task并没有本质上的不同。

handlers:

- name: restart memcached

service: name=memcached state=restarted

- name: restart apache

service: name=apache state=restarted

**4、tags**

tags用于让用户选择运行或略过playbook中的部分代码。ansible具有幂等性，因此会自动跳过没有变化的部分，即便如此，有些代码为测试其确实没有发生变化的时间依然会非常地长。此时如果确信其没有变化就可以通过tags跳过这些代码片断。

**5、示例**

一个安装httpd web服务的示例：

# cat /etc/ansible/playbook/install\_web.yml

- hosts: webservers

remote\_user: root

gather\_fasks:**False**

vars:

packages: httpd

tasks: - name:**Install** httpd yum: name={{ packages }} state=present- name:**Cofiguration** httpd copy: src=/root/httpd.conf dest=/etc/httpd/conf/httpd.conf tags: httpd\_conf

notify: - restart httpd- name:**Start** httpd service: name=httpd state=started enabled=no tags: start- name:**Add** centos user user: name={{ item }} state=absent tags: adduser with\_items: - centos - admin handlers: - name: restart httpd service: name=httpd state=restart

**Ansible7：Playbook常用模块**

playbook的模块与在ansible命令行下使用的模块有一些不同。这主要是因为在playbook中会使用到一些facts变量和一些通过setup模块从远程主机上获取到的变量。有些模块没法在命令行下运行，就是因为它们需要这些变量。而且即使那些可以在命令行下工作的模块也可以通过playbook的模块获取一些更高级的功能。

**1、template**

在实际应用中，我们的配置文件有些地方可能会根据远程主机的配置的不同而有稍许的不同，template可以使用变量来接收远程主机上setup收集到的facts信息，针对不同配置的主机，定制配置文件。用法大致与copy模块相同。

常用参数：

backup：如果原目标文件存在，则先备份目标文件

dest：目标文件路径

force：是否强制覆盖，默认为yes

group：目标文件属组

mode：目标文件的权限

owner：目标文件属主

src：源模板文件路径

validate：在复制之前通过命令验证目标文件，如果验证通过则复制

官方简单示例：

- template: src=/mytemplates/foo.j2 dest=/etc/file.conf owner=bin group=wheel mode=0644

- template: src=/mytemplates/foo.j2 dest=/etc/file.conf owner=bin group=wheel mode="u=rw,g=r,o=r"

- template: src=/mine/sudoers dest=/etc/sudoers validate='visudo -cf %s'

named.conf配置文件的jinja2模板示例：

options {

listen-on port 53 {

127.0.0.1;

{% for ip in ansible\_all\_ipv4\_addresses %}

{{ ip }};

{% endfor %}

};

listen-on-v6 port 53 { ::1; };

directory "/var/named";

dump-file "/var/named/data/cache\_dump.db";

statistics-file "/var/named/data/named\_stats.txt";

memstatistics-file "/var/named/data/named\_mem\_stats.txt";

};

zone "." IN {

type hint;

file "named.ca";

};

include "/etc/named.rfc1912.zones";

include "/etc/named.root.key";

{# Variables for zone config #}

{% if 'authorativenames' in group\_names %}

{% set zone\_type = 'master' %}

{% set zone\_dir = 'data' %}

{% else %}

{% set zone\_type = 'slave' %}

{% set zone\_dir = 'slaves' %}

{% endif %}

zone "internal.example.com" IN {

type {{ zone\_type }};

file "{{ zone\_dir }}/internal.example.com";

{% if 'authorativenames' not in group\_names %}

masters { 192.168.2.2; };

{% endif %}

};

playbook的引用该模板配置文件的方法示例：

- name: Setup BIND

host: allnames

tasks:

- name: configure BIND

template: src=templates/named.conf.j2 dest=/etc/named.conf owner=root group=named mode=0640

**2、set\_fact**

set\_fact模块可以自定义facts，这些自定义的facts可以通过template或者变量的方式在playbook中使用。如果你想要获取一个进程使用的内存的百分比，则必须通过set\_fact来进行计算之后得出其值，并将其值在playbook中引用。

下面是一个配置mysql innodb buffer size的示例：

- name: Configure MySQL

hosts: mysqlservers

tasks:

- name: install MySql

yum: name=mysql-server state=installed

- name: Calculate InnoDB buffer pool size

set\_fact: innodb\_buffer\_pool\_size\_mb="{{ ansible\_memtotal\_mb / 2 }}"

- name: Configure MySQL

template: src=templates/my.cnf dest=/etc/my.cnf owner=root group=root mode=0644

notify: restart mysql

- name: Start MySQL

service: name=mysqld state=started enabled=yes

handlers:

- name: restart mysql

service: name=mysqld state=restarted

my.cnf的配置示例：

# {{ ansible\_managed }}

[mysqld]

datadir=/var/lib/mysql

socket=/var/lib/mysql/mysql.sock

# Disabling symbolic-links is recommended to prevent assorted

security risks

symbolic-links=0

# Configure the buffer pool

innodb\_buffer\_pool\_size = {{ innodb\_buffer\_pool\_size\_mb|int }}M

[mysqld\_safe]

log-error=/var/log/mysqld.log

pid-file=/var/run/mysqld/mysqld.pid

**3、pause**

在playbook执行的过程中暂停一定时间或者提示用户进行某些操作

常用参数：

minutes：暂停多少分钟

seconds：暂停多少秒

prompt：打印一串信息提示用户操作

示例：

- name: wait on user input

pause: prompt="Warning! Detected slight issue. ENTER to continue CTRL-C a to quit"

- name: timed wait

pause: seconds=30

**4、wait\_for**

在playbook的执行过程中，等待某些操作完成以后再进行后续操作

常用参数：

connect\_timeout：在下一个任务执行之前等待连接的超时时间

delay：等待一个端口或者文件或者连接到指定的状态时，默认超时时间为300秒，在这等待的300s的时间里，wait\_for模块会一直轮询指定的对象是否到达指定的状态，delay即为多长时间轮询一次状态。

host：wait\_for模块等待的主机的地址，默认为127.0.0.1

port：wait\_for模块待待的主机的端口

path：文件路径，只有当这个文件存在时，下一任务才开始执行，即等待该文件创建完成

state：等待的状态，即等待的文件或端口或者连接状态达到指定的状态时，下一个任务开始执行。当等的对象为端口时，状态有started，stoped，即端口已经监听或者端口已经关闭；当等待的对象为文件时，状态有present或者started，absent，即文件已创建或者删除；当等待的对象为一个连接时，状态有drained，即连接已建立。默认为started

timeout：wait\_for的等待的超时时间,默认为300秒

示例：

- wait\_for: port=8080 state=started #等待8080端口已正常监听，才开始下一个任务，直到超时

- wait\_for: port=8000 delay=10 #等待8000端口正常监听，每隔10s检查一次，直至等待超时

- wait\_for: host=0.0.0.0 port=8000 delay=10 state=drained #等待8000端口直至有连接建立

- wait\_for: host=0.0.0.0 port=8000 state=drained exclude\_hosts=10.2.1.2,10.2.1.3 #等待8000端口有连接建立，如果连接来自10.2.1.2或者10.2.1.3，则忽略。

- wait\_for: path=/tmp/foo #等待/tmp/foo文件已创建

- wait\_for: path=/tmp/foo search\_regex=completed #等待/tmp/foo文件已创建，而且该文件中需要包含completed字符串

- wait\_for: path=/var/lock/file.lock state=absent #等待/var/lock/file.lock被删除

- wait\_for: path=/proc/3466/status state=absent #等待指定的进程被销毁

- local\_action: wait\_for port=22 host="{{ ansible\_ssh\_host | default(inventory\_hostname) }}" search\_regex=OpenSSH delay=10 #等待openssh启动，10s检查一次

**5、assemble**

用于组装文件，即将多个零散的文件，合并一个大文件

常用参数：

src：原文件(即零散文件)的路径

dest：合并后的大文件路径

group：合并后的大文件的属组

owner：合并后的大文件的属主

mode：合并后的大文件的权限

validate：与template的validate相同，指定命令验证文件

ignore\_hidden：组装时，是否忽略隐藏文件，默认为no，该参数在2.0版本中新增

示例：

- hosts: all

tasks:

- name: Make a Directory in /opt

file: path=/opt/sshkeys state=directory owner=root group=root mode=0700

- name: Copy SSH keys over

copy: src=keys/{{ item }}.pub dest=/opt/sshkeys/{{ item }}.pub owner=root group=root mode=0600

with\_items:

- dan

- kate

- mal

- name: Make the root users SSH config directory

file: path=/root/.ssh state=directory owner=root group=root mode=0700

- name: Build the authorized\_keys file

assemble: src=/opt/sshkeys/ dest=/root/.ssh/authorized\_keys owner=root group=root mode=0700 #将/opt/sshkeys目录里所有的文件合并到/root/.ssh/authorized\_keys一个文件中

**6、add\_host**

在playbook执行的过程中，动态的添加主机到指定的主机组中

常用参数：

groups：添加主机至指定的组

name：要添加的主机名或IP地址

示例：

- name: add a host to group webservers

hosts: webservers

tasks:

- add\_host name={{ ip\_from\_ec2 }} group=webservers foo=42 #添加主机到webservers组中，主机的变量foo的值为42

**7、group\_by**

在playbook执行的过程中，动态的创建主机组

示例：

- name: Create operating system group

hosts: all

tasks:

- group\_by: key=os\_{{ ansible\_distribution }} #在playbook中设置一个新的主机组

- name: Run on CentOS hosts only

hosts: os\_CentOS

tasks:

- name: Install Apache

yum: name=httpd state=latest

- name: Run on Ubuntu hosts only

hosts: os\_Ubuntu

tasks:

- name: Install Apache

apt: pkg=apache2 state=latest

**8、debug**

调试模块，用于在调试中输出信息

常用参数：

msg：调试输出的消息

var：将某个任务执行的输出作为变量传递给debug模块，debug会直接将其打印输出

verbosity：debug的级别

示例：

# Example that prints the loopback address and gateway for each host- debug: msg="System {{ inventory\_hostname }} has uuid {{ ansible\_product\_uuid }}"

- debug: msg="System {{ inventory\_hostname }} has gateway {{ ansible\_default\_ipv4.gateway }}"

when: ansible\_default\_ipv4.gateway is defined

- shell: /usr/bin/uptime

register: result

- debug: var=result verbosity=2 #直接将上一条指令的结果作为变量传递给var，由debug打印出result的值

- name: Display all variables/facts known for a host

debug: var=hostvars[inventory\_hostname] verbosity=4

**9、fail**

用于终止当前playbook的执行，通常与条件语句组合使用，当满足条件时，终止当前play的运行。可以直接由failed\_when取代。

选项只有一个：

msg：终止前打印出信息

示例：

- fail: msg="The system may not be provisioned according to the CMDB status."

when: cmdb\_status != "to-be-staged"

**Ansible8：Playbook循环**

在使用ansible做自动化运维的时候，免不了的要重复执行某些操作，如：添加几个用户，创建几个MySQL用户并为之赋予权限，操作某个目录下所有文件等等。好在playbook支持循环语句，可以使得某些需求很容易而且很规范的实现。

**1、with\_items**

with\_items是playbooks中最基本也是最常用的循环语句：

tasks:

- name:**Secure** config files

file: path=/etc/{{ item }} mode=0600 owner=root group=root

with\_items:

- my.cnf

- shadow

- fstab

上面例子表示，创建三个文件分别为my.cnf、shadow、fstab

也可以将文件列表提前赋值给一个变量，然后在循环语句中调用：

with\_items: "{{ somelist }}"

使用with\_items迭代循环的变量可以是个单纯的列表，也可以是一个较为复杂 的数据结果，如字典类型：

tasks:

- name: add several users

user: name={{ item.name }} state=present groups={{ item.groups }}

with\_items:

- { name: 'testuser1', groups: 'wheel' }

- { name: 'testuser2', groups: 'root' }

**2、with\_nested嵌套循环**

示例：

tasks:

- name: give users access to multiple databases

mysql\_user: name={{ item[0] }} priv={{ item[1] }}.\*:ALL append\_privs=yes password=foo

with\_nested:

- [ 'alice', 'bob' ]

- [ 'clientdb', 'employeedb', 'providerdb' ]

item[0]是循环的第一个列表的值['alice','bob']。item[1]是第二个列表的值。表示循环创建alice和bob两个用户，并且为其赋予在三个数据库上的所有权限。

也可以将用户列表事先赋值给一个变量：

tasks:

- name: here, 'users' contains the above list of employees

mysql\_user: name={{ item[0] }} priv={{ item[1] }}.\*:ALL append\_privs=yes password=foo

with\_nested:

- "{{users}}"

- [ 'clientdb', 'employeedb', 'providerdb' ]

**3、with\_dict**

with\_dict可以遍历更复杂的数据结构：

假如有如下变量内容：

users:

alice:

name: Alice Appleworth

telephone: 123-456-7890

bob:

name: Bob Bananarama

telephone: 987-654-3210

现在需要输出每个用户的用户名和手机号：

tasks:

- name: Print phone records

debug: msg="User {{ item.key }} is {{ item.value.name }} ({{ item.value.telephone }})"

with\_dict: "{{ users }}"

4、with\_fileglob文件匹配遍历

可以指定一个目录，使用with\_fileglob可以循环这个目录中的所有文件，示例如下：

tasks:

- name:**Make** key directory

file: path=/root/.sshkeys ensure=directory mode=0700 owner=root group=root

- name:**Upload** public keys

copy: src={{ item }} dest=/root/.sshkeys mode=0600 owner=root group=root

with\_fileglob:

- keys/\*.pub

- name:**Assemble** keys into authorized\_keys file

assemble: src=/root/.sshkeys dest=/root/.ssh/authorized\_keysmode=0600 owner=root group=root

5、with\_subelement遍历子元素

假如现在需要遍历一个用户列表，并创建每个用户，而且还需要为每个用户配置以特定的SSH key登录。变量文件内容如下：

users:

- name: alice

authorized:

- /tmp/alice/onekey.pub

- /tmp/alice/twokey.pub

mysql:

password: mysql-password

hosts:

- "%"

- "127.0.0.1"

- "::1"

- "localhost"

privs:

- "\*.\*:SELECT"

- "DB1.\*:ALL"

- name: bob

authorized:

- /tmp/bob/id\_rsa.pub

mysql:

password: other-mysql-password

hosts:

- "db1"

privs:

- "\*.\*:SELECT"

- "DB2.\*:ALL"

playbook中定义如下：

- user: name={{ item.name }} state=present generate\_ssh\_key=yes

with\_items: "{{users}}"

- authorized\_key: "user={{ item.0.name }} key='{{ lookup('file', item.1) }}'"

with\_subelements:

- users

- authorized

也可以遍历嵌套的子列表：

- name: Setup MySQL users

mysql\_user: name={{ item.0.name }} password={{ item.0.mysql.password }} host={{ item.1 }} priv={{ item.0.mysql.privs | join('/') }}

with\_subelements:

- users

- mysql.hosts

**6、with\_sequence循环整数序列**

with\_sequence可以生成一个自增的整数序列，可以指定起始值和结束值，也可以指定增长步长。 参数以key=value的形式指定，format指定输出的格式。数字可以是十进制、十六进制、八进制：

- hosts: all

tasks:

# create groups

- group: name=evens state=present

- group: name=odds state=present

# create some test users

- user: name={{ item }} state=present groups=evens

with\_sequence: start=0 end=32 format=testuser%02d

# create a series of directories with even numbers for some reason

- file: dest=/var/stuff/{{ item }} state=directory

with\_sequence: start=4 end=16 stride=2 # stride用于指定步长

# a simpler way to use the sequence plugin

# create 4 groups

- group: name=group{{ item }} state=present

with\_sequence: count=4

**7、with\_random\_choice随机选择**

从列表中随机取一个值：

- debug: msg={{ item }}

with\_random\_choice:

- "go through the door"

- "drink from the goblet"

- "press the red button"

- "do nothing"

**8、do-Util循环**

示例：

- action: shell /usr/bin/foo

register: result

until: result.stdout.find("all systems go") != -1

retries: 5

delay: 10

重复执行shell模块，当shell模块执行的命令输出内容包含"all systems go"的时候停止。重试5次，延迟时间10秒。retries默认值为3，delay默认值为5。任务的返回值为最后一次循环的返回结果。

**9、循环注册变量**

在循环中使用register时，保存的结果中包含results关键字，该关键字保存模块执行结果的列表

- shell: echo "{{ item }}"

with\_items:

- one

- two

register: echo

变量echo内容如下：

{

"changed": true,

"msg": "All items completed",

"results": [

{

"changed": true,

"cmd": "echo \"one\" ",

"delta": "0:00:00.003110",

"end": "2013-12-19 12:00:05.187153",

"invocation": {

"module\_args": "echo \"one\"",

"module\_name": "shell"

},

"item": "one",

"rc": 0,

"start": "2013-12-19 12:00:05.184043",

"stderr": "",

"stdout": "one"

},

{

"changed": true,

"cmd": "echo \"two\" ",

"delta": "0:00:00.002920",

"end": "2013-12-19 12:00:05.245502",

"invocation": {

"module\_args": "echo \"two\"",

"module\_name": "shell"

},

"item": "two",

"rc": 0,

"start": "2013-12-19 12:00:05.242582",

"stderr": "",

"stdout": "two"

}

]

}

遍历注册变量的结果：

- name: Fail if return code is not 0

fail:

msg: "The command ({{ item.cmd }}) did not have a 0 return code"

when: item.rc != 0

with\_items: "{{echo.results}}"

**10、with\_together遍历数据并行集合**

示例：

- hosts: webservers

remote\_user: root

vars:

alpha: [ 'a','b','c','d']

numbers: [ 1,2,3,4 ]

tasks:

- debug: msg="{{ item.0 }} and {{ item.1 }}"

with\_together:

- "{{ alpha }}"

- "{{ numbers }}"

输出的结果为：

ok: [192.168.1.65] => (item=['a', 1]) => {

"item": [

"a",

1

],

"msg": "a and 1"

}

ok: [192.168.1.65] => (item=['b', 2]) => {

"item": [

"b",

2

],

"msg": "b and 2"

}

ok: [192.168.1.65] => (item=['c', 3]) => {

"item": [

"c",

3

],

"msg": "c and 3"

}

ok: [192.168.1.65] => (item=['d', 4]) => {

"item": [

"d",

4

],

"msg": "d and 4"

}

loop模块一般在下面的场景中使用

1. 类似的配置模块重复了多遍
2. fact是一个列表
3. 创建多个文件,然后使用assemble聚合成一个大文件
4. 使用with\_fileglob匹配特定的文件管理

**Ansible9：条件语句**

在有的时候play的结果依赖于变量、fact或者是前一个任务的执行结果，从而需要使用到条件语句。

**一、when**

有的时候在特定的主机需要跳过特定的步骤，例如在安装包的时候，需要指定主机的操作系统类型，或者是当操作系统的硬盘满了之后，需要清空文件等,可以使用when语句来做判断 。when关键字后面跟着的是python的表达式,在表达式中你能够使用任何的变量或者fact,当表达式的结果返回的是false,便会跳过本次的任务

**1、基本用法，示例：**

---

- name: **Install** VIM

hosts: all tasks:

- name:**Install** VIM via yum

yum: name=vim-enhanced state=installed

when: ansible\_os\_family =="RedHat"

- name:**Install** VIM via apt

apt: name=vim state=installed

when: ansible\_os\_family =="Debian"

- name: **Unexpected** OS family

debug: msg="OS Family {{ ansible\_os\_family }} is not supported" fail=yes

when: **not** ansible\_os\_family =="RedHat" **or** ansible\_os\_family =="Debian"

条件语句还有一种用法,它还可以让你当达到一定的条件的时候暂停下来,等待你的输入确认。一般情况下,当ansible遭遇到error时,它会直接结束运行。那其实你可以当遭遇到不是预期的情况的时候给使用pause模块,这样可以让用户自己决定是否继续运行任务：

- name: pause **for** unexpected conditions

pause: prompt="Unexpected OS"

when: ansible\_os\_family !="RedHat"

**2、在when中使用jinja2的语法，示例：**

tasks:

- command: /bin/false

register: result #将命令执行的结果传递给result变量

ignore\_errors: True #忽略错误

- command: /bin/something

when: result|failed #如果注册变量的值 是任务failed则返回true

- command: /bin/something\_else

when: result|success #如果注册变量的值是任务success则返回true

- command: /bin/still/something\_else

when: result|skipped #如果注册变量的值是任务skipped则返回true

- command: /bin/foo

when: result|changed #如果注册变量的值是任务changed则返回true

- hosts: all

user: root

vars:

epic: true

tasks: - shell: echo "This certainly is epic!" when: epic

- shell: echo "This certainly is not epic!"

when: not epic

**4、如果变量不存在，则可以通过jinja2的'defined'命令跳过，示例：**

tasks:

- shell: echo "I've got '{{ foo }}' and am not afraid to use it!"

when: foo is defined

- fail: msg="Bailing out. this play requires 'bar'"

when: bar is not defined

**5、when在循环语句中的使用方法，示例：**

tasks:

- command: echo {{ item }}

with\_items: [ 0, 2, 4, 6, 8, 10 ]

when: item > 56、在include和roles中使用when：

在include中使用的示例：- include: tasks/sometasks.yml

when: "'reticulating splines' in output"

在roles中使用的示例：- hosts: webservers

roles:

- { role: debian\_stock\_config, when: ansible\_os\_family == 'Debian' }

**二、条件导入**

有些时候，你也许想在一个Playbook中以不同的方式做事，比如说在debian和centos上安装apache，apache的包名不同，除了when语句，还可以使用下面的示例来解决：

---

- hosts: all

remote\_user: root

vars\_files:

- "vars/common.yml"

- [ "vars/{{ ansible\_os\_family }}.yml", "vars/os\_defaults.yml" ]

tasks:

- name: make sure apache is running

service: name={{ apache }} state=running很多不同的yml文件只是包含键和值，如下：

---

# for vars/CentOS.yml

apache: httpd

somethingelse: 42

如果操作系统是’CentOS’, Ansible导入的第一个文件将是’vars/CentOS.yml’,紧接着 是’/var/os\_defaults.yml’,如果这个文件不存在。而且在列表中没有找到，就会报错。 在Debian系统中，最先查看的将是’vars/Debian.yml’而不是’vars/CentOS.yml’， 如果没找到，则寻找默认文件’vars/os\_defaults.yml’。

**三、with\_first\_found**

有些时候，我们想基于不同的操作系统，选择不同的配置文件，及配置文件的存放路径，可以借助with\_first\_found来解决：

- name: template a file

template: src={{ item }} dest=/etc/myapp/foo.conf

with\_first\_found:

- files:

- {{ ansible\_distribution }}.conf

- default.conf

paths:

- search\_location\_one/somedir/

- /opt/other\_location/somedir/

**四、failed\_when**

failed\_when其实是ansible的一种错误处理机制，是由fail模块使用了when条件语句的组合效果。示例如下：

- name: this command prints FAILED when it fails

command: /usr/bin/example-command -x -y -z

register: command\_result

failed\_when: "'FAILED' in command\_result.stderr"我们也可以直接通过fail模块和when条件语句，写成如下：

- name: this command prints FAILED when it fails

command: /usr/bin/example-command -x -y -z

register: command\_result

ignore\_errors: True

- name: fail the play if the previous command did not succeed

fail: msg="the command failed"

when: "'FAILED' in command\_result.stderr"五、changed\_when

当我们控制一些远程主机执行某些任务时，当任务在远程主机上成功执行，状态发生更改时，会返回changed状态响应，状态未发生更改时，会返回OK状态响应，当任务被跳过时，会返回skipped状态响应。我们可以通过changed\_when来手动更改changed响应状态。示例如下：

- shell: /usr/bin/billybass --mode="take me to the river"

register: bass\_result

changed\_when: "bass\_result.rc != 2" #只有该条task执行以后，bass\_result.rc的值不为2时，才会返回changed状态

# this will never report 'changed' status

- shell: wall 'beep'

changed\_when: False #当changed\_when为false时，该条task在执行以后，永远不会返回changed状态

**Ansible10：Playbook的角色与包含**

当单个playbook文件越来越大的时候，我们就需要重新来组织Playbooks了。我们可以将一个大的playbook拆成若干个小的playbook文件，然后通过include的方式，在主配置文件中将这些零碎的小文件包含进来，这叫做playbook的包含。我们也可以按照一定的规则将执行的某一类型任务放在一个目录里，并在这个目录中再次对这个playbook按照tasks,handlers，files,templates,vars等类型划分成若干文件，将对应文件存放在对应的目录中，这种组织方式就叫做playbook的roles。

**一、Playbook的包含**

playbook的包含其实就是使用include关键字

**1、tasks包含**

示例1：

一个task文件foo.yml示例如下：

# possibly saved as tasks/foo.yml

- name: placeholder foo

command: /bin/foo

- name: placeholder bar

command: /bin/bar

在另一个task文件bar.yml中包含foo.yml:

tasks:

- include: tasks/foo.yml

也可以在include的时候，带入变量：

tasks:

- include: wordpress.yml user=timmy

- include: wordpress.yml user=alice

- include: wordpress.yml user=bob

通过如下方式带入变量：

tasks:

- { include: wordpress.yml, user: timmy, ssh\_keys: [ 'keys/one.txt', 'keys/two.txt' ] }

再给一个例子：

tasks:

- include: wordpress.yml

vars:

remote\_user: timmy

some\_list\_variable:

- alpha

- beta

- gamma

**2、handlers包含**

handlers包含与tasks的包含大体类似，直接给例子：

handlers1.yml内容如下：

# this might be in a file like handlers/handlers.yml

- name: restart apache

service: name=apache state=restarted

handlers.yml包含handlers1.yml示例：

handlers:

- include: handlers/handlers.yml

**3、混合包含**

include也可以用于将一个playbook导入到另一个playbook中：

- name: this is a play at the top level of a file

hosts: all

remote\_user: root

tasks:

- name: say hi

tags: foo

shell: echo "hi..."

- include: load\_balancers.yml

- include: webservers.yml

- include: dbservers.yml

**二、角色（roles）**

**1、创建role**

创建role的步骤如下：

1)创建以roles命令的目录

2)在roles目录中分别创建角色名称命名的目录，如websrvs等

3)在每个角色命名的目录中分别创建files、handlers、meta、tasks、teamplates和vars目录，用不到的目录可以创建为空目录，也可以不创建。

4)在playbook文件中，调用各角色

roles文件组织结构示例：

group\_vas/

site.yml

webservers.yml

roles/

common/

files/

templates/

tasks/

handlers/

vars/

defaults/

meta/

webservers/

files/

templates/

tasks/

handlers/

vars/

defaults/

meta/

roles各目录的作用及可用的文件：

files：存放由copy或script等模块调用的文件

tempaltes：Jinja2模板文件

tasks：至少应该包含一个名为main.yml的文件，其定义了此角色的任务列表，些文件可以使用include包含其它的位于此目录中的task文件

handlers：至少包含一个main.yml文件，用于定义此角色用到的各handler，在handler中使用include包含的其他handler文件也应该位于此目录

vars：应当包含一个main.yml文件，用于定义此角色用到的变量

meta：应当包含一个main.yml文件，用于定义此角色的特殊设定及依赖关系等

default：为当前角色设定默认变量时使用些目录，包含一个main.yml文件

**2、引用roles**

基本引用的方法：

- hosts: webservers

roles:

- common

- webserver

也可以通过如下方法引用时带入变量：

- hosts: webservers

roles:

- common

- { role: foo\_app\_instance, dir: '/opt/a', port: 5000 }

- { role: foo\_app\_instance, dir: '/opt/b', port: 5001 }

还可以在引用时使用条件语句：

- hosts: webservers

roles:

- { role: some\_role, when: "ansible\_os\_family == 'RedHat'" }

下面也是一个带入变量的示例：

- hosts: webservers

roles:

- role: database

database\_name: {{ db\_name }}

database\_user: {{ db\_pass }}

- role: webserver

live\_hostname: web1

domains:

- example.com

- www.example.com

**3、pre\_tasks和post\_tasks**

如果在执行一个role时，需要在其前或其后依然要执行某些任务，我们可以使用pre\_tasks及post\_tasks来声明。pre\_tasks是在role之前执行，而post\_tasks则在role之后执行：

- name: deply webservers

host: webservers

vars\_files:

- secrets.yml

pre\_tasks:

- name: update yum cache

yum: update\_cache=yes

roles:

- role: apache

database\_host: {{ hostvars.db.ansible\_eth0.ipv4.address }}

domains:

- exampel.com

- [www.example.com](http://www.example.com/)

post\_tasks:

- name: print something

shell: echo "The roles have been updated!"

**4、role的依赖**

如果当前role在执行前需要依赖另一个role，我们可以在roles的meta目录中的main.yml中定义role的依赖关系。

示例1：

#roles/webservers/meta/main.yml

dependencies:

- { role: common, some\_parameter: 3 }

- { role: apache, port: 80 }

- { role: postgres, dbname: blarg, other\_parameter: 12 }

示例2：

dependencies:

- {role: ntp, ntp\_server=ntp.ubuntu.com}

- {role: web}

- {role: memcached}

**5、Ansible Galaxy**

ansible-galaxy是一个工具，我们可以利用它快速的创建一个标准的roles目录结构，还可以通过它在https:/galaxy.ansible.com上下载别人写好的roles，直接拿来用。

通过ansible-galaxy初始化一个roles的目录结构，方法如下：

ansible-galaxy init /etc/ansible/roles/websrvs

安装别人写好的roles：

ansible-galaxy install -p /etc/ansible/roles bennojoy.mysql

列出已安装的roles：

ansible-galaxy list

查看已安装的roles信息：

ansible-galaxy info bennojoy.mysql

卸载roles：

ansible-galaxy remove bennojoy.mysql

**ansible playbook使用总结**

原贴：http://yijiu.blog.51cto.com/433846/1665831

playbooks概念

task:

任务：使用各模块所执行的特性操作任务，比如：-m user -a 'name= password='

Variables:

变量：

Templates:

模板

在定义模板后可以实现各节点对应的变量来取代，表达式自身会根据当前节点所赋值做运算，之后生成的值则赋予这个参数，用于生产不同配置的配置文件，所以模板主要实现配置不同场景文本文件

而且这种使用模板语言来定义

模板语言中可以根据定义替换成特定主机的某些值

handlers:

处理器：

如果某一次操作配置文件发生改变的话，远程和本地配置文件不一致，那么肯定是将其配置文件需要进行覆盖，这样需要在配置文件发送改变的话需要触动某些机制进行操作

说白了就是事先可以在某些条件下，比如配置发生改变）被触发的操作

Rolse:

角色：层次型组织playbooks以及其所依赖的各种资源的一种机制；其角色可以单独被调用

彼此之间的server服务器的配置肯定是完全不一样的，如果对于这些节点同时配置并提供同样的playbooks 是不可能的，所以将其每个server分成组

之间有些角色或者功能是相同的，那么比如web server 之间可能有同步时间等，某些服务是通用的，因此有了这些机制之后能够实现按需要将某个类特定服务器相同操作定义基础配置，将特定功能定义成一类操作

playbook是基于YAML语言所定义的声明试配置文件

**数据结构包括：**

每个横线所表示引导一个列表中的一向，通过缩进所表示具有相同缩进的元素是同一个列表的元素

如下所示，定义playbook的结构化部分：

- name : deploy web server #定义一个任务模块，跟上这个名称

user : foouser #使用user模块定义其使用的用户是foouer

sodu : True #是否允许sudo权限

hosts : all #对所有的主机模块生效

tasks: #将一个任务定义应用某一个具体操作

- name : install apache #定义名称为install apache

apt : pkg=apache2-mpm-prefork state=latest #使用apt模块进行安装 状态为latest

首先来定义一个httpd服务，并让其能够开机自动启动服务

[root@testA test]# pwd

/opt/test

首先定义头部信息

- name: install httpd service

user: root

hosts: a

#再定义任务信息

tasks:

- name: install #定义第一个任务名称并调用yum模块，与命令一致

yum: name=httpd state=present

- name: service on #定义第二个任务

service: name=httpd enabled=yes state=started

执行

[root@testA test]# ansible-playbook web.yaml

PLAY [install httpd service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GATHERING FACTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.101.243]

ok: [192.168.101.242]

TASK: [install] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.101.243]

changed: [192.168.101.242]

TASK: [service on] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.101.243]

changed: [192.168.101.242]

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

192.168.101.242 : ok=3 changed=2 unreachable=0 failed=0

192.168.101.243 : ok=3 changed=2 unreachable=0 failed=0

ok = 3 有3个设定执行成功，分别是1.安装 2.修改服务启动状态 3.开启服务

以上为playbook的最简单的使用

**YAML语法**

YAML的语法和其他高阶语言类似，并且可以简单表达清单、散列表、标量等数据结构。其结构（Structure）通过空格来展示，序列（Sequence）里的项用"-"来代表，Map里的键值对用":"分隔。下面是一个示例。

name: John Smith

age: 41

gender: Male

spouse:

name: Jane Smith

age: 37

gender: Female

children:

- name: Jimmy Smith

age: 17

gender: Male

- name: Jenny Smith

age 13

gender: Female

**YAML最关键的部分为列表 字段 和映射**

列表的所有元素均使用“-”打头，例如：

# A list of tasty fruits

- Apple

- Orange

- Strawberry

- Mango

一定要做好缩进，通过空格进行排序，排列整齐的认为是同一级别的

**Ansible基础元素**

**facts**

在套用某个主机之前先获取其各种相关数据的一种机制

**获取某个节点的变量信息：**

[root@testA wangchao]# ansible 192.168.101.242 -m setup

直接返回了许多数据，这里所描述的功能表示基于setup所获取指定主机各种变量

在跟对方节点通信之前可以先将对方节点返回一些可用信息，而这些信息都是通过facts所获取的

而这些信息都是变量 都是可以被调用的，

有些变量很有用，比如：

"ansible\_os\_family": "RedHat",

"ansible\_pkg\_mgr": "yum",

根据反馈的结果再来决定使用哪些组件来管理

可以通过这些东西来判断来应用哪些任务

获得对方的线程，无非是cpu个数x2，那么在变量中也有体现，如下所示：

"ansible\_processor\_cores": 1,

**变量的种类**

1.自定义变量

2.facts

3.主机/invertory变量 host variables

4.组变量，将多个主机定义为一个组，如下所示：

[a]

hostip

[a:vars]

ntp\_server=ntp.xxx.com

组中可以嵌套一个组

[apache]

host\_1

host\_2

[nginx]

ngx1.xx.com

[webservers:children]

nginx

apache

[webserver:vars]

ntp\_server=ntp.xxx.com

因此常见变量有以上4类

**根据之前的反馈来判断任务是否执行成功**

获取到某一指定的结果状态才会运行，比如：

tasks:

-command: /bin/false

register: result

ingnore\_errors: True

第二种方式：

- command:/bin/something

when: result|failed #可以根据此前执行结果做出执行

when中可以判断此前某个结果是否是所期望的，进而根据其做出判断的

**迭代：**

还可以实现遍历循环操作，如下所示：

在主机组a中，添加3个用户

- name: add user

remote\_user: root

hosts: a

tasks:

- name: add serveral users

user: name={{items}} state=present

with\_items:

- test4

- test5

- test6

这里变量必须是以{{item}}进行标记，而调用变量必须是with\_items:进行赋值

**让各自执行安装，但是要使用yum安装，并且只有对方包管理器是yum**

需要用到变量判断的功能

涉及到：when

首先需要使用setup获取对方变量，如果使用的匹配yum，那么再进行操作

[root@testA ~]# ansible 192.168.101.242 -m setup

涉及变量：

"ansible\_pkg\_mgr": "yum",

涉及参数：

when: ansible\_pkg\_mgr == "yum"

当变量为yum的时候才执行操作

- name: install zsh

remote\_user: root

hosts: a

tasks:

- name: install zsh

yum: name=zsh state=present

**when: ansible\_pkg\_mgr == "yum"**

执行：

[root@testA wangchao]# ansible-playbook zsh.yaml

PLAY [install zsh] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GATHERING FACTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.101.243] #手机各个节点的facts信息

ok: [192.168.101.242]

TASK: [install zsh] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#开始执行任务

changed: [192.168.101.243]

changed: [192.168.101.242]

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

#返回的执行结果

192.168.101.242 : ok=2 changed=1 unreachable=0 failed=0

192.168.101.243 : ok=2 changed=1 unreachable=0 failed=0

**补充：在使用with\_items的时候还可以使用hashes**

user:name={{item,name}} state=present groups={{item.group}}

with\_items:

-{name:'test1',gourp: 'wheel'}

-{name:'test2',gourp: 'root'}

通过以上的操作，简单来说就是由多个列表，其主要功能就是将事先归并的主机，先通过tasks定义好的角色

联合起来共同完成一个任务

**关于状态码**

如果命令或脚本的退出状态码不为0，那么可以使用一下方式代替

tasks:

- name: run this xxx

shell: /usr/bin/somecommand || /bin/true

#或：

ignore\_errors: True

**playbook基础组建**

**handlers 触发操作**

handlers用于当关注的资源发生变化时采取一定的操作

就是说只有触发的时候才会操作

使用**notify**这个action可以用于每个play的最后被触发，这样可以避免多次发生改变每次都执行指定的操作

所以可以指定每次什么时候出发handlers

官方给出的例子：

- name: template config file

template: src=template.j2 dest=/etc/foo.conf #万一这个配置文件发生了改变，那么执行下面操作

notify:

- restart service #这里的restart是需要定义handlers的,所以要保持一致

- restart service(httpd)

handlers:

- name restart memcache

service: name=memcache state=restarted

- name: restart httpd

service: name=httpd state=restarted

以上意思为，如果src的配置文件发生了改变，那么则触发这个handlers定义的操作

接下来我们定义一个httpd的配置文件，当文件发生改变的时候，则推送这个配置文件到各个节点并重启httpd服务

[root@testA test]# mkdir httpd

[root@testA test]# cd httpd/

[root@testA httpd]# pwd

/opt/test/httpd

[root@testA httpd]# cp /etc/httpd/conf/httpd.conf .

[root@testA httpd]# ls

httpd.conf

为了演示效果将端口监听在8080端口

Listen 8080

效果如下：

[root@testA httpd]# cat httpd.yaml

- name: conf

remote\_user: root

hosts: a

tasks:

- name: install httpd

yum: name=httpd state=present

when: ansible\_pkg\_mgr == "yum"

- name: conf file

copy: src=/opt/test/httpd/httpd.conf dest=/etc/httpd/conf/httpd.conf

notify: restart httpd

- name: start httpd

service: name=httpd enabled=yes state=started

handlers:

- name: restart httpd

service: name=httpd state=restarted

以上为从检测对端是否为yum包管理器 如果是yum则安装httpd 再cp配置文件到指定目录最后将服务加到开机启动项并启动，最后重启服务

但是以上有几个步骤重复了

**改进:**

如果第二次执行的时候就没有必要去重新yum安装

在playbook中 其中一个选项 -t

-t表示可以执行某些特定标签对应的tasks

接下来为每个步骤都添加一个标签，如下所示：

- name: conf file

copy: src=/opt/test/httpd/httpd.conf dest=/etc/httpd/conf/httpd.conf

tags: conf

notify: restart httpd

执行：

[root@testA httpd]# ansible-playbook httpd.yaml -t conf

PLAY [conf] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GATHERING FACTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.101.243]

ok: [192.168.101.242]

TASK: [conf file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.101.243]

ok: [192.168.101.242]

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

192.168.101.242 : ok=2 changed=0 unreachable=0 failed=0

192.168.101.243 : ok=2 changed=0 unreachable=0 failed=0

如果是多个任务，也可以同时使用一个tags

都在每个tasks 指定tags: conf 即可

**通过目录结构的方式也将其一组资源再次切分**

如下所示：

直接使用inculde进行调用其他的yaml文件

[root@testA httpd]# mkdir tasks

将以下部分保留

[root@testA httpd]# vim tasks/main.yaml

- name: install httpd

yum: name=httpd state=present

when: ansible\_pkg\_mgr == "yum"

- name: conf file

copy: src=/opt/test/httpd/httpd.conf dest=/etc/httpd/conf/httpd.conf

tags: conf

notify: restart httpd #用来调用handlers

- name: start httpd

service: name=httpd enabled=yes state=started

编辑httpd.yaml

[root@testA httpd]# cat httpd.yaml

- name: conf

remote\_user: root

hosts: a

tasks:

include: tasks/main.yaml

**定义handlers**

[root@testA httpd]# mkdir handlers

[root@testA httpd]# cd handlers/

[root@testA handlers]# vim main.yaml

内容如下：

- name: restart httpd

service: name=httpd state=restarted

通过以上的定义，再修改主配置文件：

[root@testA httpd]# pwd

/opt/test/httpd

[root@testA httpd]# cat httpd.yaml

- name: conf

remote\_user: root

hosts: a

tasks:

include: tasks/main.yaml

handlers:

include: handlers/main.yaml

执行操作并查看

如果没有问题则改一下httpd.conf配置文件并指定标签

[root@testA httpd]# ansible-playbook httpd.yaml -t conf

**一个roles的案例如下所示：**

site.yml

webservers.yml

fooservers.yml

roles/

common/

files/

templates/

tasks/

handlers/

vars/

meta/

webservers/

files/

templates/

tasks/

handlers/

vars/

meta/

创建role的步骤

(1) 创建以roles命名的目录；

(2) 在roles目录中分别创建以各角色名称命名的目录，如webservers等；

(3) 在每个角色命名的目录中分别创建files、handlers、meta、tasks、templates和vars目录；用不到的目录可以创建为空目录，也可以不创建；

(4) 在playbook文件中，调用各角色；

**rolse**

再定义一个目录rolse

[root@testA httpd]# mkdir roles

将目录的文件都挪到roles里

[root@testA httpd]# mv files/ handlers/ httpd.yaml tasks/ roles/web/

[root@testA httpd]# ls roles/

web

这里的roles 表示角色目录，而这个角色名就叫做web

而在这个角色当中，通常命名为site.yaml

[root@testA web]# mv httpd.yaml site.yaml

定义一个yaml 用于调用site.yaml

[root@testA httpd]# cat webserver.yaml

- name: web service

remote\_user: root

hosts: a

roles:

- web

**Ansible11：变量详解**

**一、在Inventory中定义变量**

详见《Ansible2：主机清单》

**二、在Playbook中定义变量**

**1、通过vars关键字定义：**

vars:

http\_port: 80

server\_name: localhost

cert\_file: /etc/nginx/ssl/nginx.crt

key\_file: /etc/nginx/ssh/nginx.key

conf\_file: /etc/nginx/conf/default.conf

**2、通过vars\_files关键字引入变量文件：**

- hosts: all

remote\_user: root

vars:

favcolor: blue

vars\_files:

- /vars/external\_vars.yml

- /vars/nginx\_vars.yml

/vars/nginx\_vars.yml示例：

http\_port: 80

server\_name: localhost

cert\_file: /etc/nginx/ssl/nginx.crt

key\_file: /etc/nginx/ssh/nginx.key

conf\_file: /etc/nginx/conf/default.conf

**3、通过vars\_prompt来实现人机交互：**

hosts: all

remote\_user: root

vars\_prompt:

- name: 'https\_passphrase' #存储数据的变量名

prompt: 'Key Passphrase' #手工输入数据

private: yes #当该值为yes，则用户的输入不会被打印

**4、通过playbook的roles定义变量**

详见《ansible10：Playbook的角色与包含》

**三、注册变量**

在有些时候，我们希望把某一条任务执行的结果保存下来，可以在接下的任务中调用或者做些判断，可以通过register关键字来实现：下面是个简单的例子，如果/etc/motd文件中包含有'hi'字符串时，则输出"mothd contains ther word hi"：- name: test play

hosts: all

tasks:

- shell: cat /etc/motd

register: motd\_contents

- shell: echo "motd contains the word hi"

when: motd\_contents.stdout.find('hi') != -1下面是一个register的变量在循环中使用的例子：- 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: path=/mnt/bkspool/{{ item }} src=/home/{{ item }} state=link

with\_items: home\_dirs.stdout\_lines

# same as with\_items: home\_dirs.stdout.split()

**四、通过fact获取远程主机变量**

我们在之前讲ad-hoc常用模块的时候提到setup模块，用于获取远程主机的相关信息，并可以将这些信息作为变量在playbook里进行调用。而setup模块获取这些信息的方法就是依赖于fact。在这里，我们不再详细说明获取到的默认fact的内容。ansible除了能获取到预定义的fact的内容,还支持手动为某个主机定制fact。称之为本地fact。本地fact默认存放于目标主机的/etc/ansible/facts.d目录下，如果文件为.ini格式或者json格式，ansible会自动识别。以这种形式加载的fact是key为ansible\_local的特殊变量。

下面是一个简单的示例，一个.ini格式的example.fact文件内容如下：

[book]

title=Ansible Book

author=Breeze Yan

将其复制到目标主机的/etc/ansible/facts.d/目录，通过debug模块打印输出：

- name: print ansible\_local

debug: var=ansibl\_local

会打印出如下内容：

ok: [localhost] => {

"var": {

"ansible\_local": {

"example": {

"book": {

"author": "Breeze Yan",

"title": "Ansible Book"

}

}

}

}

}

如果不想从fact中获取变量，可以通过如下方法关闭fact：

- hosts: whatever

gather\_facts: no

**五、使用set\_fact模块定义新的变量**

在《ansible7：Playbook常用模块》中有对set\_fact模块用法的详细说明

**六、内置变量**

**1、hostvars**

获取某台指定的主机的相关变量。如果有一台web服务器的配置文件中需要指定db服务器的ip地址，我们假定这台db服务器的hostname为db.exmaple.com,ip地址绑定在eth0网卡上，我们可以通过如下方法在web服务器上调用db服务器的ip地址：

{{ hostvars['db.example.com'].ansible\_eth0.ipv4.address }}

需要注意的是db.example.com不能使用ip地址来取代，只能使用主机名或别名。

**2、inventory\_hostname与inventory\_hostname\_short**

inventory\_hostname是Ansible所识别的当前正在运行task的主机的主机名。如果在inventory里定义过别名，那么这里就是那个别名，如果inentory包含如下一行：

server1 ansible\_ssh\_host=192.168.1.1

则inventory\_hostname即为server1

利用hostvars和inventory\_hostname变量，可以输出与当前主机相关联的所有变量：

- debug: var=hostvars[inventory\_hostname]

与inventory\_hostname相近的还有一个inventory\_hostname\_short，如果一台主机的inventory\_hostname为server1.exmaple.com，则inventory\_hostname\_short的值为server1

**3、group\_names**

用于标识当前正在执行task的目标主机位于的主机组。假如我们有三台主机，用来配置成一主二从的mysql服务器。inventory配置如下：

[mdb]

db1

[sdb]

db2

db3

mysql配置文件my.conf.j2示例如下：

{% if 'db1' in group\_names %}

[mysqld]

server-id=1

log-bin=mysql-bin

log-bin-index=mysql-bin.index

sync-binlog=1

innodb\_flush\_log\_at\_trx\_commit=1 #我们知道db1在mdb组，当db1与当前执行c一组时，我们认为当前主机即在mdb组，所以对当前主机应用mysql master的配置

{% else %}

[mysqld]

server-id=2

relay-log=relay-log

relay-log-index=relay-log.index

read-only = yes

sync\_master\_info = 1

sync\_relay\_log = 1

sync\_relay\_log\_info = 1

relay\_log\_recovery = 1

skip\_slave\_start #当db1与当前主机不在同一组时，则认为当前主机不在mdb组，即应用my slave的配置

{% endif %}

我们执行如下task:

- name: copy config file to mysql master

template: src=my.conf.j2 dest=/etc/my.cnf

**4、groups**

当你想要访问一组主机的变量时，groups变量会很有用。假如我们有一个inventory文件定义如下：

[web]

server1

server2

在配置一台HAproxy的负载均衡器时，我们的配置文件肯定需要web群组的所有服务器的IP，配置文件包含如下片段：

backend web-backend

{% for host in groups.web%}

server {{host.inventory\_hostname}} {{ host.ansible\_default\_ipv4.address }}:80

{% endfor %}

最终生成的文件如下：

backend web-backend

server server1 192.168.1.1

server server2 192.168.1.2

再给一个例子，在所有的dbservers组的服务器上创建一个数据库用户kate：

- name: Create a user for all db servers

mysql\_user: name=kate password=test host={{ hostvars.[item].ansible\_eth0.ipv4.address }} state=present

with\_items: groups['dbservers']

5、play\_hosts #当前playbook会在哪些hosts上运行

6、ansible\_version #当前ansible的版本

7、inventory\_dir #主机清单所在目录

8、inventory\_file #主机清单文件

**七、通过命令行设置变量**

示例如下：

---

- hosts: '{{ hosts }}'

remote\_user: '{{ user }}'

tasks:

- ...

ansible-playbook release.yml --extra-vars "hosts=vipers user=starbuck"

也可以写成类似如下方式：

--extra-vars '{"pacman":"mrs","ghosts":["inky","pinky","clyde","sue"]}'

为了方便调试，ansible提供了debug模块来很方便的查看模块。用法可以参考通过fact获取主机变量中的打印本地fact的示例九、变量优先级

1、extra vars(命令中-e)最优先

2、inventory 主机清单中连接变量(ansible\_ssh\_user 等)

3、play 中 vars、vars\_files 等

4、剩余的在 inventory 中定义的变量

5、系统的 facts 变量

6、角色定义的默认变量(roles/rolesname/defaults/main.yml)

注：子组会覆盖父组，主机总是覆盖组定义的变量

ansible学习笔记(简要)

**Ansible部署及使用(入门级)**

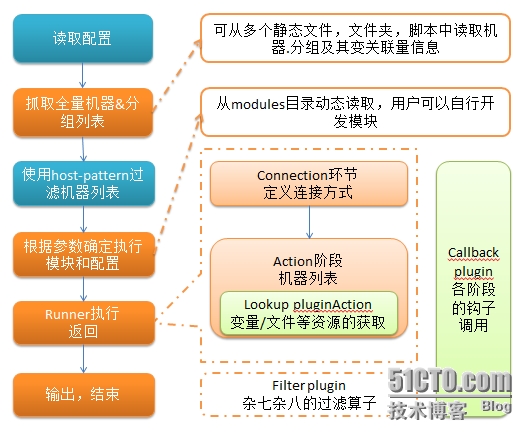
原贴：<http://shine009.blog.51cto.com/2548914/1425236>

一.什么是ansible

什么是ansible？官方的title是“Ansibleis Simple IT Automation”—>简单的自动化IT工具。ansible功能：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。其本质就是在一个台或者几台服务器上，批量的执行命令。

fabric和ansible有什么差别呢？简单来说fabric像是一个工具箱，提供了很多好用的工具，用来在Remote执行命令，而ansible则是提供了一套简单的流程，你要按照它的流程来做，就能轻松完成任务。当然，它们之间也是有共同点的——都是基于paramiko 开发的。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/agent，因为它们是基于ssh来和远程主机通讯的。

Ansible的执行过程如下图所示，暖色调带便已经模块化。



二.Ansible的安装

配置好yum，直接yum -y install ansible即可。

|  |  |
| --- | --- |
| 1 | yum -y install ansible |

安装完成后会在/etc 下生成两个文件：ansible的配置文件ansible.cfg和定义被管理节点的hosts.

三.Ansible的具体使用

Ansible Server：10.0.0.122

节点信息：

|  |
| --- |
| cat /etc/ansible/hosts 添加如下内容  [test]  10.0.0.128 #nginx01  10.0.0.129 #nginx02  10.0.0.130 #nginx03 |

建立Server及各节点的信任关系：

|  |
| --- |
| ssh-keygen -t rsa -P '' -f .ssh/id\_rsa  ssh-copy-id -i .ssh/id\_rsa.pub 10.0.0.128--10.0.0.130 |

ansible命令的用法：

建立完信任关系后，测试下各节点与Server的连通性：

|  |
| --- |
| [root@cobbler ~]# ansible all -m ping #all表示所有节点,-m指定模块 默认为command,ping指定动作  10.0.0.129 | success >> {  "changed": false,  "ping": "pong"  }  10.0.0.130 | success >> {  "changed": false,  "ping": "pong"  }  10.0.0.128 | success >> {  "changed": false,  "ping": "pong"  } |

ansible批量执行命令示例：

|  |
| --- |
| [root@cobbler ~]# ansible all -a 'uptime'  说明：#all所有节点, -a用于指定命令,‘uptime’执行的命令，如果命令有参数一定要用引号；ansible默认使用command模块，所以可以不用指定。  10.0.0.130 | success | rc=0 >>  11:31:50 up 2:01, 1 user, load average: 0.00, 0.00, 0.00  10.0.0.129 | success | rc=0 >>  11:31:50 up 2:02, 1 user, load average: 0.00, 0.00, 0.00  10.0.0.128 | success | rc=0 >>  11:31:50 up 2:05, 1 user, load average: 0.00, 0.00, 0.00    [root@cobbler ~]# ansible-doc -l #查看ansible的所有模块  acl Sets and retrieves file ACL information.  add\_host add a host (and alternatively a group) to the ansible-playbo  airbrake\_deployment Notify airbrake about app deployments  apt Manages apt-packages    [root@cobbler ~]# ansible-doc yum #查看指定模块的文档  > YUM    Installs, upgrade, removes, and lists packages and groups with the  `yum' package manager.    [root@cobbler ~]# ansible all -m yum -a "name=httpd state=present"  #用yum模块给各节点安装httpd。name指定安装包名，state指定动作：安装可以用present和latest，卸载使用absent。  10.0.0.128 | success >> {  "changed": false,  "msg": "",  "rc": 0,  "results": [  "httpd-2.2.15-30.el6.centos.x86\_64 providing httpd is already installed"  ]  }    10.0.0.130 | success >> {  "changed": false,  "msg": "",  "rc": 0,  "results": [  "httpd-2.2.15-30.el6.centos.x86\_64 providing httpd is already installed"  ]  }    10.0.0.129 | success >> {  "changed": false,  "msg": "",  "rc": 0,  "results": [  "httpd-2.2.15-30.el6.centos.x86\_64 providing httpd is already installed"  ]  } |

ansible执行shell比较麻烦，建议安装ansible-shell

ansible-shell内置的命令主要有四个：

cd : 切换到指定的组/表达式筛选的机器集合上

list： 显示目前的机器集合，list groups 可以列出所有的组（对我们可能没啥用）

serial：运行时的并发度，默认是20

help： 顾名思义，他能生成简单的模块帮助信息，方便即时查询

|  |  |
| --- | --- |
| yum -y install git  git clone https://github.com/dominis/ansible-shell.git  yum -y install python-pip  pip install -e ./ansible-shell | |
| [root@cobbler ~]# ansible-shell  Welcome to the ansible-shell.  Type help or ? to list commands.  root@ (0)[s:2]$ cd 10.0.0.128 #cd到指定host。  root@10.0.0.128 (1)[s:2]$ ls #list出了10.0.0.128这个主机上/root下的文件  =============== 10.0.0.128 ================  anaconda-ks.cfg  install.log  install.log.syslog  nginx-1.4.7  nginx-1.4.7.tar.gz  nginx\_install.sh |

其他功能还在摸索中，用到后再做补充。

**ansible安装配置与简单使用**

原贴：<http://hypocritical.blog.51cto.com/3388028/1580229>

前言：

AnsibleWorks成立于2012年，由自动化工具Cobbler及Func的开发者Michael DeHaan创建。其Ansible平台是一个开源的配置及计算机管理平台。可实现多节点的软件部署，执行特定任务并进行配置管理。

Ansible 跟其他IT自动化技术的区别在于其关注点并非配置管理、应用部署或IT流程工作流，而是提供一个统一的界面来协调所有的IT自动化功能，因此 Ansible的系统更加易用，部署更快。受管理的节点无需安装额外的远程控制软件，由平台通过SSH（Secure SHell）对其进行管理，因此十分方便。其模块支持JSON等标准输出格式，可采用任何编程语言重写。

Ansible可以让用户避免编写脚本或代码来管理应用，同时还能搭建工作流实现IT任务的自动化执行。IT自动化可以降低技术门槛及对传统IT的依赖，从而加快项目的交付速度。

ansible有如下优点：

1、轻量级，他不需要去客户端安装agent，更新时，只需要在操作机上进行一次更新即可

2、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

3、使用python编写的，维护更简单

4、支持sudo

摘抄自网络

——安装ansible

1）创建ansible用户

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | [root@node1 ~]# useradd ansible  [root@node1 ~]# passwd ansible  更改用户 ansible 的密码 。  新的 密码：  重新输入新的 密码：  passwd： 所有的身份验证令牌已经成功更新。 |

2）赋予root权限

|  |  |
| --- | --- |
| 1  2 | [root@node1 ~]# vi /etc/sudoers  ansible ALL=(ALL) NOPASSWD:ALL |

3）安装ansible

4）配置ansible

4）配置ansible

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | [root@node1 ansible-1.7.2]# vi /etc/ansible/ansible.cfg  hostfile = /etc/ansible/hosts  library = /usr/share/ansible  remote\_tmp = $HOME/.ansible/tmp  pattern = \*  forks = 5  poll\_interval = 15  sudo\_user = ansible  #ask\_sudo\_pass = True  #ask\_pass = True  transport = smart  remote\_port = 22  module\_lang = C  [root@node1 ansible-1.7.2]# vi /etc/ansible/hosts  #server  [localhost]  127.0.0.1  #client  [client]  192.168.253.129  192.168.253.130  192.168.253.131 |

5）ssh互信

6）远程ssh互信配置以及测试

——使用ansible

1）使用ping模块测试是否成功

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | [ansible@node1 ~]$ chmod g-wx,o-wx .python-eggs/  [ansible@node1 ~]$ ansible all -m ping  192.168.253.131 | success >> {  "changed": false,  "ping": "pong"  }    192.168.253.129 | success >> {  "changed": false,  "ping": "pong"  }    192.168.253.130 | success >> {  "changed": false,  "ping": "pong"  }    127.0.0.1 | success >> {  "changed": false,  "ping": "pong"  } |

2）查看时间

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | [ansible@node1 ~]$ ansible all -m command -a "sudo date"  192.168.253.131 | success | rc=0 >>  Thu Nov 20 17:50:09 CST 2014    192.168.253.129 | success | rc=0 >>  Thu Nov 20 17:50:09 CST 2014    192.168.253.130 | success | rc=0 >>  Thu Nov 20 17:50:09 CST 2014    127.0.0.1 | success | rc=0 >>  Thu Nov 20 17:50:09 CST 2014 |

3）安装软件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | #使用yum安装软件  [ansible@node1 ~]$ ansible all -m command -a "sudo yum install zip unzip -y"  192.168.253.131 | success | rc=0 >>  Loaded plugins: fastestmirror  Loading mirror speeds from cached hostfile  \* base: mirrors.yun-idc.com  \* extras: mirrors.yun-idc.com  \* updates: mirrors.yun-idc.com  Setting up Install Process  Package zip-3.0-1.el6.x86\_64 already installed and latest version  Package unzip-6.0-1.el6.x86\_64 already installed and latest version  Nothing to do  #说明此软件之前在每台服务器都已经装过了  192.168.253.129 | success | rc=0 >>  Loaded plugins: fastestmirror  Loading mirror speeds from cached hostfile  \* base: mirrors.btte.net  \* extras: mirrors.btte.net  \* updates: mirrors.yun-idc.com  Setting up Install Process  Package zip-3.0-1.el6.x86\_64 already installed and latest version  Package unzip-6.0-1.el6.x86\_64 already installed and latest version  Nothing to do    192.168.253.130 | success | rc=0 >>  Loaded plugins: fastestmirror  Loading mirror speeds from cached hostfile  \* base: mirrors.yun-idc.com  \* extras: mirrors.yun-idc.com  \* updates: mirrors.yun-idc.com  Setting up Install Process  Package zip-3.0-1.el6.x86\_64 already installed and latest version  Package unzip-6.0-1.el6.x86\_64 already installed and latest version  Nothing to do    127.0.0.1 | success | rc=0 >>  Loaded plugins: fastestmirror  Loading mirror speeds from cached hostfile  \* base: mirrors.yun-idc.com  \* extras: mirrors.yun-idc.com  \* updates: mirrors.yun-idc.com  Setting up Install Process  Package zip-3.0-1.el6.x86\_64 already installed and latest version  Package unzip-6.0-1.el6.x86\_64 already installed and latest version  Nothing to do |

4）查看ansible内置模块

[ansible@node1 ~]$ ansible-doc -l

ansible结合docker部署

**运维自动化之ansible playbook结合docker安装smokeping**

原贴：<http://dl528888.blog.51cto.com/2382721/1538444>

本次介绍ansible的paly book结合docker进行虚拟机里安装2.6.8版本smokeping（apache版本是2.4.7）。

如果大家想使用我的例子，可以从github里下载

（github地址是https://github.com/dl528888/ansible-examples/tree/master/smokeping\_install），然后放到/etc/ansible目录里，下面是内容

docker版本

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | 09:26:53 # docker version  Client version: 0.11.1  Client API version: 1.11  Go version (client): go1.2.1  Git commit (client): fb99f99/0.11.1  Server version: 0.11.1  Server API version: 1.11  Git commit (server): fb99f99/0.11.1  Go version (server): go1.2.1  Last stable version: 1.1.2, please update docker |

ansible版本

|  |  |
| --- | --- |
| 1  2 | 09:28:13 # ansible --version  ansible 1.4.3 |

1、查看docker已有镜像

|  |  |
| --- | --- |
| 1  2  3  4  5 | 09:25:55 # docker images  REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE  ubuntu 3.0 6cee55276528 9 weeks ago 369.8 MB  centos5 3.0 e08d23b09189 9 weeks ago 840.9 MB  centos6 3.0 e94a3b24a19b 9 weeks ago 415.9 MB |
|  |  |

可以看到有3个镜像，1个是ubuntu，1个centos5，1个centos6.我这里打算使用centos6来弄。

2、加载新容器

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 09:31:01 # time docker inspect $(docker run -d -p 22 -p 80:80 --name="smokeping" centos6:3.0 /usr/sbin/sshd -D)|grep -i address|awk -F '"' '{print $4}'  172.17.0.5    real 0m4.737s  user 0m0.038s  sys 0m0.054s |

可以看到4秒就加载新容器完成，并且镜像为centos6系统，容器名为smokeping，开启了ssh服务，并且暴漏了22与80端口。

下面在从后端看看本机都运行了哪些容器

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 09:31:29 # docker ps -a  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  56b70c31a07e centos6:3.0 /usr/sbin/sshd -D About a minute ago Up About a minute 0.0.0.0:80->80/tcp, 0.0.0.0:49156->22/tcp smokeping  846efb9e4d7a ubuntu:3.0 /usr/sbin/sshd -D 12 days ago Up 4 days 0.0.0.0:49167->22/tcp ubuntu-test1  b9a9e6f2caed centos6:3.0 /usr/sbin/sshd -D 3 weeks ago Up 4 days 0.0.0.0:49166->22/tcp zabbix-server  978fff134b18 centos6:3.0 /usr/sbin/sshd -D 4 weeks ago Up 4 days 0.0.0.0:49165->22/tcp centos6-test5 |

下面是ansible安装smokeping的部分

3、ansible安装smokeping的信息

|  |  |
| --- | --- |
| 1  2  3  4  5 | 09:34:06 # cat smokeping\_install/vars/main.yml  smokeping\_dir: /usr/local/smokeping  smokeping\_version: 2.6.8  smokeping\_user: admin  smokeping\_passwd: '()TF%penst\*&MedHU' |

可以看到smokeping安装目录是/usr/local/smokeping，安装版本是2.6.8，登陆的账户是admin，登陆密码是()TF%penst\*&MedHU

4、下面是安装smokeping的playbook结构

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | 09:36:25 # tree smokeping\_\*  smokeping\_delete  ├── files  │ └── delete\_smokeping.sh  ├── handlers  ├── meta  │ └── main.yml  ├── tasks  │ ├── copy.yml  │ ├── delete.yml  │ └── main.yml  ├── templates  └── vars  ├── main.xml  ├── main.yml  └── vars  └── main.yml  smokeping\_install  ├── files  │ ├── rpmforge-release-0.5.3-1.el6.rf.x86\_64.rpm  │ └── smokeping.tar.gz  ├── handlers  ├── meta  │ └── main.yml  ├── tasks  │ ├── copy.yml  │ ├── delete.yml  │ ├── install.yml  │ └── main.yml  ├── templates  │ ├── config  │ └── smokeping.conf  └── vars  └── main.yml    13 directories, 18 files |

5、playbook安装smokeping的内容是

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | 09:37:27 # cat smokeping\_install.yml  ---  - hosts: "{{host}}"  remote\_user: "{{user}}"  gather\_facts: True  roles:  - common  - pcre\_install  - apache\_install  - smokeping\_install |

6、playbook删除smokeping的内容是

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | 09:38:01 # cat smokeping\_delete.yml  ---  - hosts: "{{host}}"  remote\_user: "{{user}}"  gather\_facts: True  roles:  - apache\_delete  - pcre\_delete  - smokeping\_delete |

下面是安装与测试过程

7、把docker新容器smokeping的ip加入到ansible的hosts里

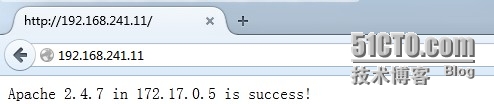
|  |  |
| --- | --- |
| 1 | echo "172.17.0.5">>/etc/ansible/hosts |

8、安装smokeping

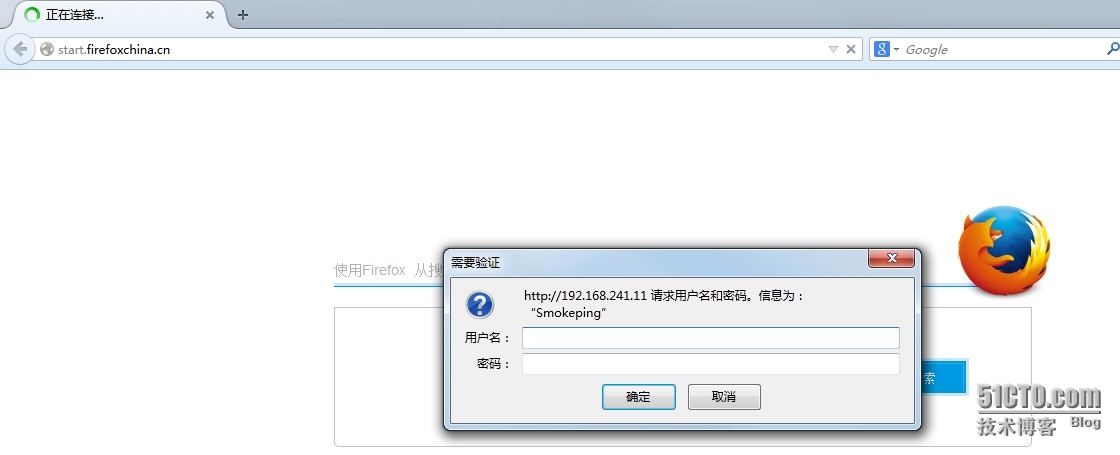
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125 | 09:39:10 # time ansible-playbook smokeping\_install.yml --extra-vars "host=172.17.0.5 user=root" -k  SSH password:    PLAY [172.17.0.5] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    GATHERING FACTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ok: [172.17.0.5]    TASK: [common | Install initializtion require software] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [pcre\_install | Copy Pcre Software To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [pcre\_install | Uncompression Pcre Software In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [pcre\_install | Delete Pcre Software In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [apache\_install | Copy Apache Software To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5] => (item=httpd-2.4.7.tar.gz)  changed: [172.17.0.5] => (item=libiconv.tar.gz)    TASK: [apache\_install | Create Apache User In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [apache\_install | Uncompression Apache Software To Redhat Client] \*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [apache\_install | Copy Apache Config To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [apache\_install | Copy Apache Vhost Config To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [apache\_install | Copy Apache Start Service Script To Redhat Client] \*\*\*  changed: [172.17.0.5]    TASK: [apache\_install | Create Lib Install Dir] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ok: [172.17.0.5]    TASK: [apache\_install | Check Apache Iconv In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [apache\_install | Install Apache Iconv In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [apache\_install | Check Lib In Config In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  failed: [172.17.0.5] => {"changed": true, "cmd": "grep -c /usr/local/lib/ /etc/ld.so.conf ", "delta": "0:00:00.006524", "end": "2014-08-11 09:40:48.822372", "item": "", "rc": 1, "start": "2014-08-11 09:40:48.815848"}  stdout: 0  ...ignoring    TASK: [apache\_install | Install Apache Iconv In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [apache\_install | Create Apache Dir] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5] => (item=/data/webroot/apache)  changed: [172.17.0.5] => (item=/data/webroot/apache/logs)  changed: [172.17.0.5] => (item=/data/webroot/apache/vhost)    TASK: [apache\_install | Install Check Script In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [apache\_install | Create Index Html To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [apache\_install | Start Apache Service In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [apache\_install | Add Boot Start Apache Service In Redhat Client] \*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [apache\_install | Delete Apache compression Software In Redhat Client] \*\*\*  changed: [172.17.0.5]    TASK: [smokeping\_install | Create Smokeping Install Dir] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [smokeping\_install | Copy Smokeping Software To Redhat Client] \*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5] => (item=rpmforge-release-0.5.3-1.el6.rf.x86\_64.rpm)  changed: [172.17.0.5] => (item=smokeping.tar.gz)    TASK: [smokeping\_install | Install Epel Yum Repo] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [smokeping\_install | Install Require Software] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [smokeping\_install | Uncompression Smokeping] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [smokeping\_install | Copy Smokeping Config To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [smokeping\_install | Copy Smokeping Httpd Config To Redhat Client] \*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [smokeping\_install | Check Boot Start In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  failed: [172.17.0.5] => {"changed": true, "cmd": "grep -c '/usr/local/smokeping/bin/smokeping --logfile=/var/log/smokeping.log 2>&1 &' /etc/rc.local ", "delta": "0:00:00.007325", "end": "2014-08-11 09:46:03.045378", "item": "", "rc": 1, "start": "2014-08-11 09:46:03.038053"}  stdout: 0  ...ignoring    TASK: [smokeping\_install | Add Boot Start In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [smokeping\_install | Modify Smokeping Dir Permission In Redhat Client] \*\*\*  changed: [172.17.0.5]    TASK: [smokeping\_install | Start Smokeping Service In Redhat Client] \*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [smokeping\_install | Restart Apache Service In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [smokeping\_install | Delete Installed File] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  172.17.0.5 : ok=35 changed=33 unreachable=0 failed=0      real 6m52.934s  user 0m15.104s  sys 0m1.632s |

花费6分52秒完成，花费时间多的主要是yum安装基础库与使用epel安装smokeping依赖库。

9、安装后测试

[](http://s3.51cto.com/wyfs02/M01/45/58/wKiom1PoInjjIePxAACBGl7w5fg476.jpg)

查看smokeping

[](http://s3.51cto.com/wyfs02/M00/45/58/wKiom1PoItTggjuhAAGrTdO6Ihw932.jpg)

由于我做了安全认证，需要输入账户与密码，这个账户与密码就上文第3步的账户与密码

[](http://s3.51cto.com/wyfs02/M00/45/58/wKiom1PoIxHQYHtkAADwWndNQgA311.jpg)

然后默认的其他网络监控（包括1、2、3线电信、移动、联通、铁通与教育网的节点监控已经做好）

[](http://s3.51cto.com/wyfs02/M01/45/59/wKioL1PoJGrjwA7CAAVKSkSX8wU360.jpg)

如果有其他的监控需要自己来添加即可。

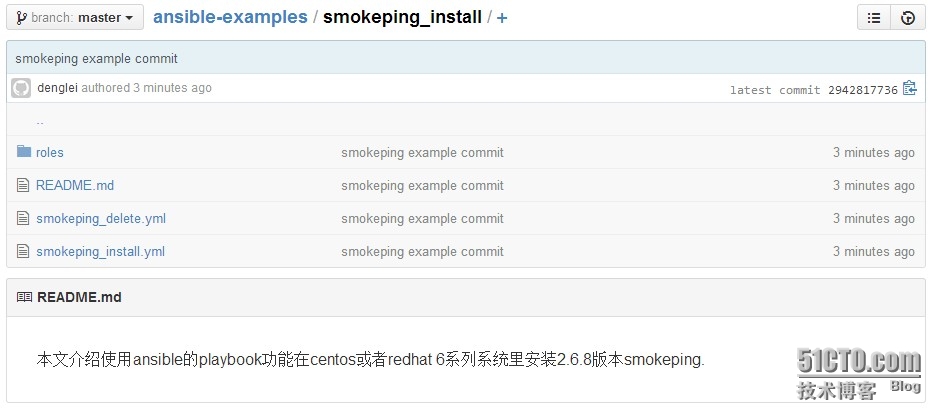
10、删除

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | 09:46:08 # time ansible-playbook smokeping\_delete.yml --extra-vars "host=172.17.0.5 user=root" -k  SSH password:    PLAY [172.17.0.5] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    GATHERING FACTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ok: [172.17.0.5]    TASK: [apache\_delete | Stop Httpd Service In RedHat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [apache\_delete | Delete Boot Start In RedHat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [apache\_delete | Delete Apache Dir In RedHat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [apache\_delete | Delete Apache Service Script In RedHat Client] \*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [apache\_delete | Delete Apache User] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [pcre\_delete | Delete Pcre] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [smokeping\_delete | Stop Smokeping Service In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [smokeping\_delete | Delete Smokeping Dir In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    TASK: [smokeping\_delete | Delete Boot Start In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [172.17.0.5]    PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  172.17.0.5 : ok=10 changed=9 unreachable=0 failed=0      real 0m9.326s  user 0m2.914s  sys 0m0.446s |

11、删除后测试

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | 09:59:45 # ssh 172.17.0.5  root@172.17.0.5's password:  root@56b70c31a07e:~  09:59:57 # ifconfig  eth0 Link encap:Ethernet HWaddr A6:80:57:2D:D3:F1  inet addr:172.17.0.5 Bcast:0.0.0.0 Mask:255.255.0.0  inet6 addr: fe80::a480:57ff:fe2d:d3f1/64 Scope:Link  UP BROADCAST RUNNING MTU:1500 Metric:1  RX packets:84520 errors:0 dropped:0 overruns:0 frame:0  TX packets:86385 errors:0 dropped:0 overruns:0 carrier:0  collisions:0 txqueuelen:1000  RX bytes:69075188 (65.8 MiB) TX bytes:9400662 (8.9 MiB)    lo Link encap:Local Loopback  inet addr:127.0.0.1 Mask:255.0.0.0  inet6 addr: ::1/128 Scope:Host  UP LOOPBACK RUNNING MTU:16436 Metric:1  RX packets:2 errors:0 dropped:0 overruns:0 frame:0  TX packets:2 errors:0 dropped:0 overruns:0 carrier:0  collisions:0 txqueuelen:0  RX bytes:168 (168.0 b) TX bytes:168 (168.0 b)    root@56b70c31a07e:~  09:59:59 # ps -ef|grep httpd  root 1017 1002 0 10:00 pts/0 00:00:00 grep httpd  root@56b70c31a07e:~  10:00:04 # ps -ef|grep smokeping  root 1019 1002 0 10:00 pts/0 00:00:00 grep smokeping  root@56b70c31a07e:~  10:00:07 # ll /usr/local/  total 40  drwxr-xr-x 2 root root 4096 Sep 23 2011 bin  drwxr-xr-x 2 root root 4096 Sep 23 2011 etc  drwxr-xr-x 2 root root 4096 Sep 23 2011 games  drwxr-xr-x 2 root root 4096 Sep 23 2011 include  drwxr-xr-x 2 root root 4096 Aug 11 09:40 lib  drwxr-xr-x 2 root root 4096 Sep 23 2011 lib64  drwxr-xr-x 2 root root 4096 Sep 23 2011 libexec  drwxr-xr-x 2 root root 4096 Sep 23 2011 sbin  drwxr-xr-x 5 root root 4096 Sep 23 2011 share  drwxr-xr-x 2 root root 4096 Sep 23 2011 src  root@56b70c31a07e:~  10:00:11 # cat /etc/rc.local  #!/bin/sh  #  # This script will be executed \*after\* all the other init scripts.  # You can put your own initialization stuff in here if you don't  # want to do the full Sys V style init stuff.    touch /var/lock/subsys/local |

如果大家想使用我的例子，可以从github里下载（地址是https://github.com/dl528888/ansible-examples/tree/master/smokeping\_install），然后放到/etc/ansible目录里，下面是内容

[](http://s3.51cto.com/wyfs02/M01/45/58/wKiom1PoJXfDxRdDAAHarjz6Fe0814.jpg)

本文出自 “[吟—技术交流](http://dl528888.blog.51cto.com/)” 博客，请务必保留此出处<http://dl528888.blog.51cto.com/2382721/1538444>

**运维自动化之ansible playbook安装mysql**

原贴：<http://dl528888.blog.51cto.com/2382721/1437882>

上次介绍了如何使用ansible playbook安装zabbix客户端（http://dl528888.blog.51cto.com/2382721/1436745），这次介绍一下如何使用playbook安装mysql。

下面是安装mysql的信息：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | mysql\_basedir: /data/mysql/basedir 源码目录  mysql\_datadir: /data/mysql/datadir 数据目录  mysql\_user: mysql mysql用户  mysql\_database\_user: root 数据库用户  mysql\_passwd: 'E4yR3WnoluSFTCBAI' 数据库密码  mysql\_port: 3306 mysql监听端口  mysql\_sock: /data/mysql/datadir/mysql.sock mysql的sock  mysql\_charset: utf8 mysql字符集  mysql\_collation: utf8\_general\_ci mysql排序方式  mysql\_version: Percona-Server-5.5.21-rel25.1.tar.gz mysql版本 |

下面是安装mysql的playbook结构

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | 10:24:06 # tree mysql\_\*  mysql\_delete  ├── files  ├── handlers  ├── meta  │ └── main.yml  ├── tasks  │ ├── delete.yml  │ └── main.yml  ├── templates  └── vars  └── main.yml  mysql\_install  ├── files  │ └── mysql.tar.gz  ├── handlers  ├── meta  │ └── main.yml  ├── tasks  │ ├── copy.yml  │ ├── delete.yml  │ ├── install.yml  │ └── main.yml  ├── templates  │ ├── install\_mysql.sh  │ ├── my.cnf  │ ├── mysqld  │ └── mysql\_security.sh  └── vars  └── main.yml    12 directories, 15 files |

playbook的mysql安装的是

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | 10:32:06 # cat mysql\_install.yml  ---  - hosts: "{{host}}"  remote\_user: "{{user}}"  gather\_facts: True  roles:  - common  - mysql\_install |

playbook的mysql协助的是

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | 10:32:44 # cat mysql\_delete.yml  ---  - hosts: "{{host}}"  remote\_user: "{{user}}"  gather\_facts: True  roles:  - mysql\_delete |

由于在上篇安装zabbix客户端已经介绍了结构，所以现在不介绍了。

1、安装mysql

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57 | 10:26:00 # time ansible-playbook mysql\_install.yml --extra-vars "host=192.168.240.17 user=root" --private-key=/root/test.pem    PLAY [192.168.240.17] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    GATHERING FACTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ok: [192.168.240.17]    TASK: [common | Install initializtion require software] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [mysql\_install | Copy Mysql Software To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [mysql\_install | Create Mysql User In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [mysql\_install | Copy Mysql Start Script To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [mysql\_install | Copy Install Mysql Script To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [mysql\_install | Copy Mysql Config To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [mysql\_install | Copy Mysql Security Script To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [mysql\_install | Create Mysql Install Dir] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ok: [192.168.240.17]    TASK: [mysql\_install | Uncompression Mysql Software To Redhat Client] \*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [mysql\_install | Modify Mysql Dir Permission In Redhat Client] \*\*\*\*\*\*\*\*\*\*  ok: [192.168.240.17] => (item=/data/mysql/datadir)  ok: [192.168.240.17] => (item=/data/mysql/basedir)    TASK: [mysql\_install | Install Mysql Script In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [mysql\_install | Start Myql Security Script In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [mysql\_install | Add Boot Start Mysql Service In Redhat Client] \*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [mysql\_install | Delete Mysql compression Software In Redhat Client] \*\*\*\*  changed: [192.168.240.17]    PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  192.168.240.17 : ok=15 changed=12 unreachable=0 failed=0      real 2m1.596s  user 0m8.815s  sys 0m0.848s |

2、安装后测试

登陆192.168.240.17的测试机查看mysql安装情况

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60 | [root@ip-10-10-240-21 tmp]# ll  total 16  lrwxrwxrwx 1 root root 30 Jul 13 22:27 mysql.sock -> /data/mysql/datadir/mysql.sock  -rw------- 1 root root 256 Jul 10 06:07 tmp.0PLkgCq81n  -rw------- 1 root root 197 Jul 9 05:35 yum\_save\_tx-2014-07-09-09-35ibcBiO.yumtx  -rw-rw-r-- 1 zabbix zabbix 320 Jul 10 21:39 zabbix\_agentd.log  -rw-rw-r-- 1 zabbix zabbix 5 Jul 10 21:39 zabbix\_agentd.pid  [root@ip-10-10-240-21 tmp]# ps -ef|grep mysql  root 21333 1 0 22:27 ? 00:00:00 /bin/sh /data/mysql/basedir/bin/mysqld\_safe --defaults-file=/data/mysql/datadir/my.cnf --datadir=/data/mysql/datadir --pid-file=/data/mysql/datadir/mysql.pid  mysql 22156 21333 2 22:27 ? 00:00:08 /data/mysql/basedir/bin/mysqld --defaults-file=/data/mysql/datadir/my.cnf --basedir=/data/mysql/basedir --datadir=/data/mysql/datadir --plugin-dir=/data/mysql/basedir/lib/plugin --user=mysql --log-error=/data/mysql/datadir/mysql-error.log --open-files-limit=10240 --pid-file=/data/mysql/datadir/mysql.pid --socket=/data/mysql/datadir/mysql.sock --port=3306  root 22440 20233 0 22:33 pts/0 00:00:00 grep mysql  [root@ip-10-10-240-21 tmp]# ll /data/mysql/  total 8  drwxr-xr-x 13 mysql mysql 4096 Jun 26 03:59 basedir  drwxr-xr-x 4 mysql mysql 4096 Jul 13 22:28 datadir  [root@ip-10-10-240-21 tmp]# ll /etc/init.d/mysqld  -rwxr-xr-x 1 root root 10905 Jul 13 22:27 /etc/init.d/mysqld  [root@ip-10-10-240-21 tmp]# /etc/init.d/mysqld status  MySQL (Percona Server) running (22156) [ OK ]  [root@ip-10-10-240-21 tmp]# chkconfig --list|grep mysql  mysqld 0:off 1:off 2:on 3:on 4:on 5:on 6:off  [root@ip-10-10-240-21 tmp]# mysql -h 192.168.240.17 -u root -pE4yR3WnoluSFTCBAI  ERROR 1045 (28000): Access denied for user 'root'@'192.168.240.17' (using password: YES)  [root@ip-10-10-240-21 tmp]# mysql -h 10.10.240.21 -u root -pE4yR3WnoluSFTCBAI  Welcome to the MySQL monitor. Commands end with ; or \g.  Your MySQL connection id is 8  Server version: 5.5.21-log Source distribution    Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.    Oracle is a registered trademark of Oracle Corporation and/or its  affiliates. Other names may be trademarks of their respective  owners.    Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.      mysql> show databases;  +--------------------+  | Database |  +--------------------+  | information\_schema |  | mysql |  | performance\_schema |  +--------------------+  3 rows in set (0.00 sec)    mysql> use mysql  Reading table information for completion of table and column names  You can turn off this feature to get a quicker startup with -A    Database changed  mysql> select host,user,password from user;  +--------------+--------+-------------------------------------------+  | host | user | password |  +--------------+--------+-------------------------------------------+  | 10.10.240.21 | root | \*BE78618CBAFFF409CE17D81579C1678B94439BE1 |  | % | zabbix | \*DEEF4D7D88CD046ECA02A80393B7780A63E7E789 |  +--------------+--------+-------------------------------------------+  2 rows in set (0.00 sec) |

可以看到已经按照我的要求安装完成，下面在进行删除测试。

3、删除

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | 10:49:18 # time ansible-playbook mysql\_delete.yml --extra-vars "host=192.168.240.17 user=root" --private-key=/root/test.pem    PLAY [192.168.240.17] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    GATHERING FACTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ok: [192.168.240.17]    TASK: [mysql\_delete | Stop Mysql Service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [mysql\_delete | Delete Mysql Boot Start Script] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [mysql\_delete | Delete Mysql Dir And Socket] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [mysql\_delete | Delete Mysql User] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [mysql\_delete | Delete Mysql Service Start Script] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  192.168.240.17 : ok=6 changed=5 unreachable=0 failed=0      real 0m25.248s  user 0m0.632s  sys 0m0.102s |

4、删除后测试

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | [root@ip-10-10-240-21 tmp]# ll /tmp/  total 16  -rw------- 1 root root 256 Jul 10 06:07 tmp.0PLkgCq81n  -rw------- 1 root root 197 Jul 9 05:35 yum\_save\_tx-2014-07-09-09-35ibcBiO.yumtx  -rw-rw-r-- 1 zabbix zabbix 320 Jul 10 21:39 zabbix\_agentd.log  -rw-rw-r-- 1 zabbix zabbix 5 Jul 10 21:39 zabbix\_agentd.pid  [root@ip-10-10-240-21 tmp]# ll /data/  total 4  drwxr-xr-x 3 root root 4096 Jul 10 22:11 webroot  [root@ip-10-10-240-21 tmp]# ps -ef|grep mysql  root 22955 20233 0 22:50 pts/0 00:00:00 grep mysql  [root@ip-10-10-240-21 tmp]# ll /etc/init.d/mysql  ls: cannot access /etc/init.d/mysql: No such file or directory  [root@ip-10-10-240-21 tmp]# chkconfig --list|grep mysql  [root@ip-10-10-240-21 tmp]# |

可以看到都删除完成。

如果大家想使用我的例子，可以从github里下载（地址是https://github.com/dl528888/ansible-examples/tree/master/mysql\_install），然后放到/etc/ansible目录里，下面是压缩包里的内容

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47 | 11:20:08 # unzip -v mysql\_install.zip  Archive: mysql\_install.zip  Length Method Size Cmpr Date Time CRC-32 Name  -------- ------ ------- ---- ---------- ----- -------- ----  113 Defl:N 89 21% 07-14-2014 10:32 aaed0763 mysql\_install.yml  99 Defl:N 82 17% 07-14-2014 10:32 70c2a028 mysql\_delete.yml  0 Stored 0 0% 07-03-2014 17:29 00000000 roles/common/  0 Stored 0 0% 06-30-2014 15:58 00000000 roles/common/meta/  267 Defl:N 178 33% 06-30-2014 15:58 31ee20ec roles/common/meta/main.yml  0 Stored 0 0% 07-11-2014 09:31 00000000 roles/common/tasks/  586 Defl:N 268 54% 07-08-2014 10:18 1c0af2a1 roles/common/tasks/main.yml  0 Stored 0 0% 06-19-2014 13:43 00000000 roles/common/handlers/  58 Defl:N 48 17% 06-19-2014 13:43 8d058053 roles/common/handlers/main.yml  0 Stored 0 0% 06-19-2014 13:30 00000000 roles/common/vars/  0 Stored 0 0% 06-19-2014 13:30 00000000 roles/common/templates/  0 Stored 0 0% 06-19-2014 13:30 00000000 roles/common/files/  0 Stored 0 0% 06-24-2014 11:27 00000000 roles/mysql\_install/  0 Stored 0 0% 07-08-2014 14:12 00000000 roles/mysql\_install/meta/  198 Defl:N 149 25% 07-08-2014 14:12 b03e00eb roles/mysql\_install/meta/main.yml  0 Stored 0 0% 07-14-2014 09:27 00000000 roles/mysql\_install/tasks/  201 Defl:N 153 24% 07-14-2014 09:27 51de730b roles/mysql\_install/tasks/delete.yml  65 Defl:N 41 37% 07-14-2014 09:27 a3197ca7 roles/mysql\_install/tasks/main.yml  1281 Defl:N 329 74% 07-14-2014 09:27 73d60454 roles/mysql\_install/tasks/copy.yml  1162 Defl:N 375 68% 07-14-2014 09:27 e5a7341c roles/mysql\_install/tasks/install.yml  0 Stored 0 0% 06-19-2014 13:30 00000000 roles/mysql\_install/handlers/  0 Stored 0 0% 07-14-2014 09:51 00000000 roles/mysql\_install/vars/  313 Defl:N 177 44% 07-14-2014 09:51 bf4476a7 roles/mysql\_install/vars/main.yml  0 Stored 0 0% 07-14-2014 11:11 00000000 roles/mysql\_install/templates/  518 Defl:N 241 54% 07-14-2014 09:15 529052bf roles/mysql\_install/templates/install\_mysql.sh  10905 Defl:N 3765 66% 06-26-2014 14:06 e35e7b82 roles/mysql\_install/templates/mysqld  2122 Defl:N 823 61% 07-14-2014 09:49 f91c74ab roles/mysql\_install/templates/my.cnf  794 Defl:N 275 65% 07-03-2014 13:15 560ee63e roles/mysql\_install/templates/mysql\_security.sh  0 Stored 0 0% 06-26-2014 13:59 00000000 roles/mysql\_install/files/  61805835 Defl:N 60413846 2% 06-26-2014 16:18 f61e317d roles/mysql\_install/files/mysql.tar.gz  0 Stored 0 0% 06-24-2014 11:27 00000000 roles/mysql\_delete/  0 Stored 0 0% 07-08-2014 14:12 00000000 roles/mysql\_delete/meta/  197 Defl:N 147 25% 07-08-2014 14:12 fe6b0ef8 roles/mysql\_delete/meta/main.yml  0 Stored 0 0% 07-14-2014 09:39 00000000 roles/mysql\_delete/tasks/  692 Defl:N 245 65% 07-14-2014 09:39 dccd57b4 roles/mysql\_delete/tasks/delete.yml  22 Stored 22 0% 06-27-2014 13:30 728ce4aa roles/mysql\_delete/tasks/main.yml  0 Stored 0 0% 06-19-2014 13:30 00000000 roles/mysql\_delete/handlers/  0 Stored 0 0% 07-03-2014 13:16 00000000 roles/mysql\_delete/vars/  313 Defl:N 177 44% 07-14-2014 10:56 bf4476a7 roles/mysql\_delete/vars/main.yml  0 Stored 0 0% 06-27-2014 13:30 00000000 roles/mysql\_delete/templates/  0 Stored 0 0% 06-27-2014 13:30 00000000 roles/mysql\_delete/files/  -------- ------- --- -------  61825741 60421430 2% 41 files |

如果觉得好，给个赞并多评论，谢谢。

**运维自动化之ansible playbook安装nginx**

原贴：<http://dl528888.blog.51cto.com/2382721/1438847>

上次介绍了如何使用ansible playbook安装mysql（http://dl528888.blog.51cto.com/2382721/1437882），这次介绍一下如何使用playbook安装nginx。

下面是安装nginx的信息：

|  |
| --- |
| nginx\_user: www nginx用户  nginx\_port: 80 nginx监听端口  nginx\_web\_dir: /data/webroot/nginx nginx的目录  nginx\_version: 1.4.3 nginx的版本  pcre\_version: 8.33 pcre的版本  pretended\_name: Tengine 为了安全起见，把nginx名字改为tengine  pretended\_version: 1.5.2 隐藏版本，换个tengine的版本 |

备注：此playbook仅能对centos或者redhat的6.x版本进行安装。

下面是安装nginx的playbook结构

|  |
| --- |
| 09:33:15 # tree nginx\_\*  nginx\_delete  ├── files  ├── handlers  ├── meta  │ └── main.yml  ├── tasks  │ ├── delete.yml  │ └── main.yml  ├── templates  └── vars  └── main.yml  nginx\_install  ├── files  │ └── nginx-1.4.3.tar.gz  ├── handlers  ├── meta  │ └── main.yml  ├── tasks  │ ├── copy.yml  │ ├── delete.yml  │ ├── install.yml  │ └── main.yml  ├── templates  │ ├── index.html  │ ├── index.php  │ ├── install\_nginx.sh  │ ├── nginx  │ ├── nginx.conf  │ └── vhost.conf  └── vars  └── main.yml    12 directories, 17 files |

playbook的nginx安装是

|  |
| --- |
| 09:34:04 # cat nginx\_install.yml  ---  - hosts: "{{host}}"  remote\_user: "{{user}}"  gather\_facts: True  roles:  - common  - pcre\_install  - nginx\_install |

playbook的nginx删除为

|  |
| --- |
| 09:35:06 # cat nginx\_delete.yml  ---  - hosts: "{{host}}"  remote\_user: "{{user}}"  gather\_facts: True  roles:  - nginx\_delete  - pcre\_delete |

1、安装nginx

|  |
| --- |
| 09:26:58 # time ansible-playbook nginx\_install.yml --extra-vars "host=192.168.240.17 user=root" --private-key=/root/test.pem    PLAY [192.168.240.17] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    GATHERING FACTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ok: [192.168.240.17]    TASK: [common | Install initializtion require software] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [pcre\_install | Copy Pcre Software To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [pcre\_install | Uncompression Pcre Software In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [pcre\_install | Delete Pcre Software In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [nginx\_install | Copy Nginx Software To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [nginx\_install | Uncompression Nginx Software To Redhat Client] \*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [nginx\_install | Copy Nginx Start Script To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [nginx\_install | Copy Nginx Config To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [nginx\_install | Copy Nginx Vhost Config to RedHat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [nginx\_install | Create Nginx User In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [nginx\_install | Create Nginx Dir] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17] => (item=vhost)  changed: [192.168.240.17] => (item=logs)    TASK: [nginx\_install | Modify Nginx Dir Permission In Redhat Client] \*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17] => (item=/data/webroot/nginx)  changed: [192.168.240.17] => (item=/usr/local/nginx-1.4.3)    TASK: [nginx\_install | Create Index Html To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [nginx\_install | Install Check Script In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [nginx\_install | Start Nginx Service In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [nginx\_install | Add Boot Start Nginx Service In Redhat Client] \*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [nginx\_install | Delete Nginx compression Software In Redhat Client] \*\*\*\*  changed: [192.168.240.17]    PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  192.168.240.17 : ok=18 changed=17 unreachable=0 failed=0      real 0m53.357s  user 0m2.433s  sys 0m0.341s |

2、安装后测试

|  |
| --- |
| [root@ip-10-10-240-21 local]# ps -ef|grep nginx  root 27791 1 0 21:27 ? 00:00:00 nginx: master process /usr/local/nginx-1.4.3/sbin/nginx -c /usr/local/nginx-1.4.3/conf/nginx.conf  www 27792 27791 0 21:27 ? 00:00:00 nginx: worker process  www 27794 27791 0 21:27 ? 00:00:00 nginx: worker process  www 27795 27791 0 21:27 ? 00:00:00 nginx: worker process  www 27796 27791 0 21:27 ? 00:00:00 nginx: worker process  www 27797 27791 0 21:27 ? 00:00:00 nginx: worker process  www 27798 27791 0 21:27 ? 00:00:00 nginx: worker process  www 27799 27791 0 21:27 ? 00:00:00 nginx: worker process  www 27800 27791 0 21:27 ? 00:00:00 nginx: worker process  root 27905 26349 0 21:28 pts/0 00:00:00 grep nginx  [root@ip-10-10-240-21 local]# chkconfig --list|grep nginx  nginx 0:off 1:off 2:off 3:on 4:on 5:on 6:off  [root@ip-10-10-240-21 local]# /etc/init.d/nginx status  Usage: /etc/init.d/nginx {start|restart|reload|stop|test|show}  [root@ip-10-10-240-21 local]# curl 10.10.240.21  Nginx 1.4.3 in 10.10.240.21 is success!  [root@ip-10-10-240-21 local]# curl -I 10.10.240.21  HTTP/1.1 200 OK  Server: Tengine/1.5.2  Date: Wed, 16 Jul 2014 01:35:59 GMT  Content-Type: text/html  Content-Length: 41  Last-Modified: Wed, 16 Jul 2014 01:27:57 GMT  Connection: keep-alive  ETag: "53c5d51d-29"  Accept-Ranges: bytes |

可与看到安装成功，并且软件名与版本都隐藏了

3、删除nginx

|  |
| --- |
| 09:28:02 # time ansible-playbook nginx\_delete.yml --extra-vars "host=192.168.240.17 user=root" --private-key=/root/test.pem    PLAY [192.168.240.17] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    GATHERING FACTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ok: [192.168.240.17]    TASK: [nginx\_delete | stop nginx service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [nginx\_delete | Delete Nginx Boot Start Script] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [nginx\_delete | Delete Nginx Dir] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [nginx\_delete | Delete Nginx User] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [nginx\_delete | Delete Nginx Service Start Script] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [pcre\_delete | Delete Pcre] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  192.168.240.17 : ok=7 changed=6 unreachable=0 failed=0      real 0m30.222s  user 0m0.774s  sys 0m0.126s |

4、删除后测试

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | [root@ip-10-10-240-21 local]# ps -ef|grep nginx  root 28417 26349 0 21:37 pts/0 00:00:00 grep nginx  [root@ip-10-10-240-21 local]# chkconfig --list|grep nginx  [root@ip-10-10-240-21 local]# ll /etc/init.d/nginx  ls: cannot access /etc/init.d/nginx: No such file or directory  [root@ip-10-10-240-21 local]# ll /usr/local/  total 44  drwxr-xr-x. 2 root root 4096 Sep 23 2011 bin  drwxr-xr-x. 2 root root 4096 Sep 23 2011 etc  drwxr-xr-x. 2 root root 4096 Sep 23 2011 games  drwxr-xr-x. 2 root root 4096 Sep 23 2011 include  drwxr-xr-x. 2 root root 4096 Jul 10 22:43 lib  drwxr-xr-x. 2 root root 4096 Sep 23 2011 lib64  drwxr-xr-x. 2 root root 4096 Sep 23 2011 libexec  drwxr-xr-x. 2 root root 4096 Sep 23 2011 sbin  drwxr-xr-x. 5 root root 4096 May 12 2013 share  drwxr-xr-x. 3 root root 4096 May 12 2013 src  drwxr-xr-x 5 zabbix zabbix 4096 Jul 8 02:17 zabbix |

可与看到都删除了。

如果大家想使用我的例子，可以从github里下载（地址是https://github.com/dl528888/ansible-examples/tree/master/nginx\_install），然后放到/etc/ansible目录里，下面是内容

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49 | 09:39:13 # unzip -v nginx\_install.zip  Archive: nginx\_install.zip  Length Method Size Cmpr Date Time CRC-32 Name  -------- ------ ------- ---- ---------- ----- -------- ----  132 Defl:N 95 28% 07-16-2014 09:34 d75c7ef9 nginx\_install.yml  117 Defl:N 89 24% 07-16-2014 09:35 27eb35d2 nginx\_delete.yml  0 Stored 0 0% 07-03-2014 17:29 00000000 roles/common/  0 Stored 0 0% 06-30-2014 15:58 00000000 roles/common/meta/  267 Defl:N 178 33% 07-16-2014 09:25 31ee20ec roles/common/meta/main.yml  0 Stored 0 0% 07-14-2014 13:40 00000000 roles/common/tasks/  586 Defl:N 268 54% 07-16-2014 09:25 1c0af2a1 roles/common/tasks/main.yml  0 Stored 0 0% 06-19-2014 13:43 00000000 roles/common/handlers/  58 Defl:N 48 17% 07-16-2014 09:25 8d058053 roles/common/handlers/main.yml  0 Stored 0 0% 06-19-2014 13:30 00000000 roles/common/vars/  0 Stored 0 0% 06-19-2014 13:30 00000000 roles/common/templates/  0 Stored 0 0% 06-19-2014 13:30 00000000 roles/common/files/  0 Stored 0 0% 06-24-2014 11:27 00000000 roles/nginx\_install/  0 Stored 0 0% 07-08-2014 14:12 00000000 roles/nginx\_install/meta/  198 Defl:N 149 25% 07-16-2014 09:25 e527ea95 roles/nginx\_install/meta/main.yml  0 Stored 0 0% 07-03-2014 17:46 00000000 roles/nginx\_install/tasks/  198 Defl:N 153 23% 07-16-2014 09:25 eb70a996 roles/nginx\_install/tasks/delete.yml  65 Defl:N 41 37% 07-16-2014 09:25 a3197ca7 roles/nginx\_install/tasks/main.yml  1142 Defl:N 300 74% 07-16-2014 09:25 34ddc722 roles/nginx\_install/tasks/copy.yml  1574 Defl:N 431 73% 07-16-2014 09:25 13c2a951 roles/nginx\_install/tasks/install.yml  0 Stored 0 0% 06-19-2014 13:30 00000000 roles/nginx\_install/handlers/  0 Stored 0 0% 06-24-2014 10:38 00000000 roles/nginx\_install/vars/  155 Defl:N 105 32% 07-16-2014 09:25 70e58924 roles/nginx\_install/vars/main.yml  0 Stored 0 0% 07-16-2014 09:25 00000000 roles/nginx\_install/templates/  738 Defl:N 337 54% 07-16-2014 09:25 bd02415e roles/nginx\_install/templates/vhost.conf  784 Defl:N 346 56% 06-24-2014 11:37 6d09dba5 roles/nginx\_install/templates/install\_nginx.sh  1642 Defl:N 698 58% 07-16-2014 09:25 60b4ed9a roles/nginx\_install/templates/nginx.conf  196 Defl:N 147 25% 07-16-2014 09:25 fa0f7181 roles/nginx\_install/templates/index.php  77 Defl:N 71 8% 07-16-2014 09:25 1fdd8439 roles/nginx\_install/templates/index.html  926 Defl:N 459 50% 07-16-2014 09:25 4aa2e8fa roles/nginx\_install/templates/nginx  0 Stored 0 0% 07-03-2014 14:24 00000000 roles/nginx\_install/files/  2346820 Defl:N 2283881 3% 07-16-2014 09:25 8a501ba6 roles/nginx\_install/files/nginx-1.4.3.tar.gz  0 Stored 0 0% 06-23-2014 14:03 00000000 roles/nginx\_delete/  0 Stored 0 0% 07-08-2014 14:12 00000000 roles/nginx\_delete/meta/  197 Defl:N 147 25% 07-16-2014 09:25 ab72e486 roles/nginx\_delete/meta/main.yml  0 Stored 0 0% 07-01-2014 17:05 00000000 roles/nginx\_delete/tasks/  893 Defl:N 289 68% 07-16-2014 09:25 68681d99 roles/nginx\_delete/tasks/delete.yml  22 Stored 22 0% 07-16-2014 09:25 728ce4aa roles/nginx\_delete/tasks/main.yml  0 Stored 0 0% 06-24-2014 14:14 00000000 roles/nginx\_delete/handlers/  0 Stored 0 0% 06-24-2014 12:48 00000000 roles/nginx\_delete/vars/  155 Defl:N 105 32% 07-16-2014 09:25 70e58924 roles/nginx\_delete/vars/main.yml  0 Stored 0 0% 06-19-2014 17:32 00000000 roles/nginx\_delete/templates/  0 Stored 0 0% 06-24-2014 13:53 00000000 roles/nginx\_delete/files/  -------- ------- --- -------  2356942 2288359 3% 43 files |

如果觉得好，给个赞并多评论，谢谢。

**运维自动化之ansible playbook安装lnmp环境**

原贴：<http://dl528888.blog.51cto.com/2382721/1440775>

之前介绍了使用ansible playbok安装zabbix客户端、mysql服务端、nginx，现在介绍一下安装lnmp环境。

如果大家想使用我的例子，可以从github里下载（地址是https://github.com/dl528888/ansible-examples/tree/master/lnmp\_install），然后放到/etc/ansible目录里

[](http://s3.51cto.com/wyfs02/M01/3F/B7/wKioL1PMeluAMkeYAAHxwM1hkkM273.jpg)

下面是安装nginx的信息：

|  |
| --- |
| nginx\_user: www nginx用户  nginx\_port: 80 nginx监听端口  nginx\_web\_dir: /data/webroot/nginx nginx的目录  nginx\_version: 1.4.3 nginx的版本  pcre\_version: 8.33 pcre的版本  pretended\_name: Tengine 为了安全起见，把nginx名字改为tengine  pretended\_version: 1.5.2 隐藏版本，换个tengine的版本 |

下面是安装php的信息：

|  |
| --- |
| php\_user: www php fpm模式的启动用户  php\_port: 9000 php fpm使用tcp的端口  php\_dir: /usr/local php的安装目录  php\_version: 5.3.10 php的版本  mysql\_basedir: /data/mysql/basedir mysql的目录 |

下面是安装mysql的信息：

|  |
| --- |
| mysql\_basedir: /data/mysql/basedir 源码目录  mysql\_datadir: /data/mysql/datadir 数据目录  mysql\_user: mysql mysql用户  mysql\_database\_user: root 数据库用户  mysql\_passwd: 'E4yR3WnoluSFTCBAI' 数据库密码  mysql\_port: 3306 mysql监听端口  mysql\_sock: /data/mysql/datadir/mysql.sock mysql的sock  mysql\_charset: utf8 mysql字符集  mysql\_collation: utf8\_general\_ci mysql排序方式  mysql\_version: Percona-Server-5.5.21-rel25.1.tar.gz mysql版本 |

备注：此playbook仅能对centos或者redhat的6.x版本进行安装。

下面是安装lnmp的playbook结构

|  |
| --- |
| 09:28:50 # tree nginx\* mysql\* php\*  nginx\_delete  ├── files  ├── handlers  ├── meta  │ └── main.yml  ├── tasks  │ ├── delete.yml  │ └── main.yml  ├── templates  └── vars  └── main.yml  nginx\_install  ├── files  │ └── nginx-1.4.3.tar.gz  ├── handlers  ├── meta  │ └── main.yml  ├── tasks  │ ├── copy.yml  │ ├── delete.yml  │ ├── install.yml  │ └── main.yml  ├── templates  │ ├── index.html  │ ├── index.php  │ ├── install\_nginx.sh  │ ├── nginx  │ ├── nginx.conf  │ └── vhost.conf  └── vars  └── main.yml  mysql\_delete  ├── files  ├── handlers  ├── meta  │ └── main.yml  ├── tasks  │ ├── delete.yml  │ └── main.yml  ├── templates  └── vars  └── main.yml  mysql\_install  ├── files  │ └── mysql.tar.gz  ├── handlers  ├── meta  │ └── main.yml  ├── tasks  │ ├── copy.yml  │ ├── delete.yml  │ ├── install.yml  │ └── main.yml  ├── templates  │ ├── install\_mysql.sh  │ ├── my.cnf  │ ├── mysqld  │ └── mysql\_security.sh  └── vars  └── main.yml  php\_delete  ├── files  ├── handlers  ├── meta  │ └── main.yml  ├── tasks  │ ├── delete.yml  │ └── main.yml  ├── templates  │ └── delete\_php.sh  └── vars  └── main.yml  php\_install  ├── files  │ └── php.tar.gz  ├── handlers  ├── meta  │ └── main.yml  ├── tasks  │ ├── copy.yml  │ ├── delete.yml  │ ├── install.yml  │ └── main.yml  ├── templates  │ ├── install\_php.sh  │ ├── php-fpm  │ └── php-fpm.conf  └── vars  └── main.yml    36 directories, 47 files |

playbook的lnmp安装是

|  |
| --- |
| 09:29:59 # cat lnmp\_install.yml  ---  - hosts: "{{host}}"  remote\_user: "{{user}}"  gather\_facts: True  roles:  - common  - pcre\_install  - nginx\_install  - mysql\_install  - php\_install |

playbook的lnmp删除是

|  |
| --- |
| 09:30:33 # cat lnmp\_delete.yml  ---  - hosts: "{{host}}"  remote\_user: "{{user}}"  gather\_facts: True  roles:  - pcre\_delete  - php\_delete  - nginx\_delete  - mysql\_delete |

1、安装lnmp

|  |
| --- |
| 09:35:55 # time ansible-playbook lnmp\_install.yml --extra-vars "host=192.168.240.13 user=root" --private-key=/root/test.pem    PLAY [192.168.240.13] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    GATHERING FACTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ok: [192.168.240.13]    TASK: [common | Install initializtion require software] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [pcre\_install | Copy Pcre Software To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [pcre\_install | Uncompression Pcre Software In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [pcre\_install | Delete Pcre Software In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [nginx\_install | Copy Nginx Software To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [nginx\_install | Uncompression Nginx Software To Redhat Client] \*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [nginx\_install | Copy Nginx Start Script To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [nginx\_install | Copy Nginx Config To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [nginx\_install | Copy Nginx Vhost Config to RedHat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [nginx\_install | Create Nginx User In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [nginx\_install | Create Nginx Dir] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13] => (item=vhost)  changed: [192.168.240.13] => (item=logs)    TASK: [nginx\_install | Modify Nginx Dir Permission In Redhat Client] \*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13] => (item=/data/webroot/nginx)  changed: [192.168.240.13] => (item=/usr/local/nginx-1.4.3)    TASK: [nginx\_install | Create Index Html To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [nginx\_install | Install Check Script In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [nginx\_install | Start Nginx Service In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [nginx\_install | Add Boot Start Nginx Service In Redhat Client] \*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [nginx\_install | Delete Nginx compression Software In Redhat Client] \*\*\*\*  changed: [192.168.240.13]    TASK: [mysql\_install | Copy Mysql Software To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [mysql\_install | Create Mysql User In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [mysql\_install | Copy Mysql Start Script To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [mysql\_install | Copy Install Mysql Script To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [mysql\_install | Copy Mysql Config To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [mysql\_install | Copy Mysql Security Script To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [mysql\_install | Create Mysql Install Dir] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ok: [192.168.240.13]    TASK: [mysql\_install | Uncompression Mysql Software To Redhat Client] \*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [mysql\_install | Modify Mysql Dir Permission In Redhat Client] \*\*\*\*\*\*\*\*\*\*  ok: [192.168.240.13] => (item=/data/mysql/datadir)  ok: [192.168.240.13] => (item=/data/mysql/basedir)    TASK: [mysql\_install | Install Mysql Script In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [mysql\_install | Start Myql Security Script In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [mysql\_install | Add Boot Start Mysql Service In Redhat Client] \*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [mysql\_install | Delete Mysql compression Software In Redhat Client] \*\*\*\*  changed: [192.168.240.13]    TASK: [php\_install | Copy Php Software To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [php\_install | Uncompression Php Software To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [php\_install | Copy Php Start Script To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [php\_install | Copy Php Config To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [php\_install | Copy Php Install Script To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [php\_install | Create Php User In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ok: [192.168.240.13]    TASK: [php\_install | Install Php Script In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [php\_install | Start Php Service In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [php\_install | Add Boot Start Php Service In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [php\_install | Delete Php compression Software In Redhat Client] \*\*\*\*\*\*\*\*  changed: [192.168.240.13]    PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  192.168.240.13 : ok=41 changed=37 unreachable=0 failed=0      real 2m16.803s  user 0m14.753s  sys 0m1.468s |

可以看到2分16秒就安装成功，一般比较费时的是使用yum安装软件与从本地负责包到客户端，所以yum源最后使用自己做的yum源或者找个速度快的yum源。

2、安装后测试

|  |
| --- |
| [root@ip-10-10-240-20 tmp]# ps -ef|grep nginx|wc -l  10  [root@ip-10-10-240-20 tmp]# ps -ef|grep mysql|wc -l  3  [root@ip-10-10-240-20 tmp]# ps -ef|grep php|wc -l  130  [root@ip-10-10-240-20 tmp]# cd /data/webroot/nginx/  logs/ vhost/  [root@ip-10-10-240-20 tmp]# cd /data/webroot/nginx/vhost/  [root@ip-10-10-240-20 vhost]# cat index.html  Nginx 1.4.3 in 10.10.240.20 is success!  [root@ip-10-10-240-20 vhost]# cat index.php  <?php  $link = mysql\_connect('10.10.240.20','zabbix','zabbix');  if ($link)  echo "Successs!!!Php install success!\n";  else  echo "Fail!!!Php install fail!\n";  mysql\_close();  ?>  [root@ip-10-10-240-20 vhost]# curl 10.10.240.20/index.html  Nginx 1.4.3 in 10.10.240.20 is success!  [root@ip-10-10-240-20 vhost]# curl 10.10.240.20/index.php  Successs!!!Php install success!  [root@ip-10-10-240-20 vhost]# curl -I 10.10.240.20/index.php  HTTP/1.1 200 OK  Server: Tengine/1.5.2  Date: Mon, 21 Jul 2014 02:12:24 GMT  Content-Type: text/html  Connection: keep-alive  Vary: Accept-Encoding |

3、删除lnmp

|  |
| --- |
| 09:38:36 # time ansible-playbook lnmp\_delete.yml --extra-vars "host=192.168.240.13 user=root" --private-key=/root/test.pem    PLAY [192.168.240.13] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    GATHERING FACTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ok: [192.168.240.13]    TASK: [pcre\_delete | Delete Pcre] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [php\_delete | Copy Delete Php Script To RedHat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [php\_delete | Run Delete Php Script In RedHat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [php\_delete | Delete Php User In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  failed: [192.168.240.13] => {"failed": true, "item": "", "name": "www", "rc": 8}  msg: userdel: user www is currently logged in    ...ignoring    TASK: [php\_delete | Delete Php Script] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [nginx\_delete | stop nginx service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [nginx\_delete | Delete Nginx Boot Start Script] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [nginx\_delete | Delete Nginx Dir] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [nginx\_delete | Delete Nginx User] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [nginx\_delete | Delete Nginx Service Start Script] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [mysql\_delete | Stop Mysql Service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [mysql\_delete | Delete Mysql Boot Start Script] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [mysql\_delete | Delete Mysql Dir And Socket] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [mysql\_delete | Delete Mysql User] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    TASK: [mysql\_delete | Delete Mysql Service Start Script] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.13]    PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  192.168.240.13 : ok=16 changed=14 unreachable=0 failed=0      real 0m31.511s  user 0m1.470s  sys 0m0.232s |

4、删除后测试

|  |
| --- |
| [root@ip-10-10-240-20 vhost]# ll /usr/local/  total 40  drwxr-xr-x. 2 root root 4096 Sep 23 2011 bin  drwxr-xr-x. 2 root root 4096 Sep 23 2011 etc  drwxr-xr-x. 2 root root 4096 Sep 23 2011 games  drwxr-xr-x. 2 root root 4096 Sep 23 2011 include  drwxr-xr-x. 2 root root 4096 Sep 23 2011 lib64  drwxr-xr-x. 2 root root 4096 Sep 23 2011 libexec  drwxr-xr-x. 2 root root 4096 Sep 23 2011 sbin  drwxr-xr-x. 5 root root 4096 May 12 2013 share  drwxr-xr-x. 3 root root 4096 May 12 2013 src  drwxr-xr-x 5 zabbix zabbix 4096 Jul 8 02:17 zabbix  [root@ip-10-10-240-20 vhost]# ps -ef|grep php  root 10016 3527 0 22:13 pts/0 00:00:00 grep php  [root@ip-10-10-240-20 vhost]# ps -ef|grep mysql  root 10018 3527 0 22:13 pts/0 00:00:00 grep mysql  [root@ip-10-10-240-20 vhost]# ps -ef|grep nginx  root 10020 3527 0 22:13 pts/0 00:00:00 grep nginx  [root@ip-10-10-240-20 vhost]# id www  id: www: No such user  [root@ip-10-10-240-20 vhost]# id mysql  id: mysql: No such user |

可与看到都删除了。

如果大家想使用我的例子，可以从github里下载（地址是https://github.com/dl528888/ansible-examples/tree/master/lnmp\_install），然后放到/etc/ansible目录里

[](http://s3.51cto.com/wyfs02/M01/3F/B7/wKioL1PMeluAMkeYAAHxwM1hkkM273.jpg)

**运维自动化之ansible playbook安装zabbix客户端**

原贴：<http://dl528888.blog.51cto.com/2382721/1436745>

之前介绍了关于ansible的安装与使用（包括模块与playbook使用，地址是http://dl528888.blog.51cto.com/2382721/1435415），今天介绍一下如何使用playbook来部署zabbix客户端。

ansible服务端的环境为centos 6.5 x86\_64系统

ansible客户端环境为centos 6.3 x86\_64系统

目前我的playbook只允许centos或redhat 6系列系统来安装zabbix客户端，并且客户端的版本是2.0.6.

下面是playbook的结构

|  |
| --- |
| 14:29:30 # pwd  /etc/ansible/roles  root@ip-10-10-10-10:/etc/ansible/roles  14:29:37 # tree zabbix\_client\_\*  zabbix\_client\_delete 删除已经安装的zabbix客户端  ├── files 存放文件的  ├── handlers 重启的东东  ├── meta galaxy\_info的信息  │ └── main.yml  ├── tasks 操作的任务流程  │ ├── delete.yml  │ └── main.yml  ├── templates 模板  └── vars 变量  └── main.yml  zabbix\_client\_install  ├── files  │ └── zabbix-2.0.6.tar.gz  ├── handlers  ├── meta  │ └── main.yml  ├── tasks  │ ├── copy.yml  │ ├── delete.yml  │ ├── install.yml  │ └── main.yml  ├── templates  │ ├── zabbix\_agentd  │ └── zabbix\_agentd.conf  └── vars  └── main.yml    12 directories, 13 files |

下面是先介绍一下安装方面zabbix\_client\_install的内容

1、galaxy\_info的信息

|  |
| --- |
| 14:32:15 # cat /etc/ansible/roles/zabbix\_client\_install/meta/main.yml  galaxy\_info:  author: Deng Lei  description: Install Zabbix Client  license: MIT  min\_ansible\_version: 1.6  platforms:  - name: CentOS  versions:  - 6  categories:  - Monitor  dependencies: [] |

2、task里的copy.xml信息

|  |
| --- |
| 14:33:35 # cat /etc/ansible/roles/zabbix\_client\_install/tasks/copy.yml  - name: Stop Exist Zabbix Client Service In Redhat Client  shell: ps -ef|grep zabbix|grep -v grep|awk '{print $2}'|xargs kill -9 >>/dev/null 2>&1  ignore\_errors: yes  when: ansible\_os\_family == "RedHat" and ansible\_lsb.major\_release|int == 6  - name: Delete Exist Zabbix Client Dir In Redhat Client  shell: rm -rf {{ zabbix\_dir }}/zabbix  ignore\_errors: yes  when: ansible\_os\_family == "RedHat" and ansible\_lsb.major\_release|int == 6  - name: Install Base Require Software In Redhat Client  yum: name={{ item }} state=latest  with\_items:  - telnet  - dmidecode  - tar  - name: Create Zabbix User In Redhat Client  user: name={{ zabbix\_user }} state=present createhome=no shell=/sbin/nologin  when: ansible\_os\_family == "RedHat" and ansible\_lsb.major\_release|int == 6  - name: Copy Zabbix Client Software To Redhat Client  copy: src=zabbix-{{ zabbix\_version }}.tar.gz dest=/tmp/zabbix-{{ zabbix\_version }}.tar.gz owner=root group=root  when: ansible\_os\_family == "RedHat" and ansible\_lsb.major\_release|int == 6  - name: Uncompression Zabbix Client Software To Redhat Client  shell: tar zxf /tmp/zabbix-{{ zabbix\_version }}.tar.gz -C {{ zabbix\_dir }}/  when: ansible\_os\_family == "RedHat" and ansible\_lsb.major\_release|int == 6  - name: Copy Zabbix Start Script To Redhat Client  template: src=zabbix\_agentd dest=/etc/init.d/zabbix\_agentd owner=root group=root mode=0755  when: ansible\_os\_family == "RedHat" and ansible\_lsb.major\_release|int == 6  - name: Copy Zabbix Config To Redhat Client  template: src=zabbix\_agentd.conf dest={{ zabbix\_dir }}/zabbix/conf/zabbix\_agentd.conf owner={{ zabbix\_user }} group={{ zabbix\_user }} mode=0644  when: ansible\_os\_family == "RedHat" and ansible\_lsb.major\_release|int == 6 |

此文件是复制对应的文件到客户端

3、task的install.yml信息

|  |
| --- |
| 14:34:26 # cat /etc/ansible/roles/zabbix\_client\_install/tasks/install.yml  - name: Modify Zabbix Dir Permission In Redhat Client  file: path={{ zabbix\_dir }}/zabbix owner={{ zabbix\_user }} group={{ zabbix\_user }} mode=0755  when: ansible\_os\_family == "RedHat" and ansible\_lsb.major\_release|int == 6  - name: Check Zabbix User Sudo Permission In Redhat Client  shell: grep "{{ zabbix\_user }}" /etc/sudoers|wc -l  register: zabbix\_sudoer  ignore\_errors: True  when: ansible\_os\_family == "RedHat" and ansible\_lsb.major\_release|int == 6  - name: Give Sudo Permission To Zabbix User In Redhat Client  shell: echo "{{ zabbix\_user }} ALL=(root) NOPASSWD:/bin/netstat, /usr/bin/omreport" >> /etc/sudoers  when: ansible\_os\_family == "RedHat" and ansible\_lsb.major\_release|int == 6 and zabbix\_sudoer|int ==0  - name: Start Zabbix Service In Redhat Client  shell: /etc/init.d/zabbix\_agentd start  when: ansible\_os\_family == "RedHat" and ansible\_lsb.major\_release|int == 6  - name: Add Boot Start Zabbix Service In Redhat Client  shell: chkconfig --level 345 zabbix\_agentd on  when: ansible\_os\_family == "RedHat" and ansible\_lsb.major\_release|int == 6 |

此文件主要是安装

4、tasks的delete.yml信息

|  |
| --- |
| 14:35:08 # cat /etc/ansible/roles/zabbix\_client\_install/tasks/delete.yml  - name: Delete Zabbix compression Software In Redhat Client  shell: rm -rf /tmp/zabbix-{{ zabbix\_version }}.tar.gz  when: ansible\_os\_family == "RedHat" and ansible\_lsb.major\_release|int == 6 |

此文件是安装完成后，删除安装前的文件

5、tasks的mail.yml

|  |
| --- |
| 14:35:37 # cat /etc/ansible/roles/zabbix\_client\_install/tasks/main.yml  - include: copy.yml  - include: install.yml  - include: delete.yml |

此文件是允许运行哪个文件

6、templates的zabbix\_agentd

|  |
| --- |
| 15:15:45 # cat /etc/ansible/roles/zabbix\_client\_install/templates/zabbix\_agentd  #!/bin/bash  #  # chkconfig: - 85 15  # description: Zabbix client script.  # processname: Zabbix  . /etc/profile  SERVICE="Zabbix agent"  DAEMON={{ zabbix\_dir }}/zabbix/sbin/zabbix\_agentd  PIDFILE=/tmp/zabbix\_agentd.pid  CONFIG={{ zabbix\_dir }}/zabbix/conf/zabbix\_agentd.conf  zabbix\_agent\_status=`ps aux|grep zabbix\_agentd.conf|grep -v grep|wc -l`  zabbix\_agent\_pid=`ps aux|grep zabbix\_agentd|grep -v grep|awk 'NR==1{print $2}'`  # Source function library.  . /etc/rc.d/init.d/functions  # Source networking configuration.  . /etc/sysconfig/network  function check()  {  if [ $? -eq 0 ];then  action $"Operating is:" /bin/true  else  action $"Operating is:" /bin/false  fi  }  case $1 in  'start')  if [ -x ${DAEMON} ]  then  $DAEMON -c $CONFIG  # Error checking here would be good...  echo "${SERVICE} started."  else  echo "Can't find file ${DAEMON}."  echo "${SERVICE} NOT started."  fi  check  ;;    'stop')  if [ -s ${PIDFILE} ]  then  if kill `cat ${PIDFILE}` >/dev/null 2>&1  then  echo "${SERVICE} terminated."  rm -f ${PIDFILE}  fi  fi  check  ;;  'restart')  /bin/bash $0 stop  sleep 5  /bin/bash $0 start  ;;    'status')  if [ $zabbix\_agent\_status -ne 0 ];then  echo "Zabbix Agentd is running ($zabbix\_agent\_pid)"  else  echo "Zabbix Agentd is not running!"  fi  ;;    \*)  echo "Usage: $0 {start|stop|status|restart}"  ;;    esac  exit 0 |

这个文件是启动客户端的脚本

7、templates的zabbix\_agentd.conf

|  |
| --- |
| 15:16:36 # cat /etc/ansible/roles/zabbix\_client\_install/templates/zabbix\_agentd.conf  # This is a config file for the Zabbix agent daemon (Unix)  # To get more information about Zabbix, visit http://www.zabbix.com    ############ GENERAL PARAMETERS #################    ### Option: PidFile  # Name of PID file.  #  # Mandatory: no  # Default:  # PidFile=/tmp/zabbix\_agentd.pid    ### Option: LogFile  # Name of log file.  # If not set, syslog is used.  #  # Mandatory: no  # Default:  # LogFile=    LogFile=/tmp/zabbix\_agentd.log    ### Option: LogFileSize  # Maximum size of log file in MB.  # 0 - disable automatic log rotation.  #  # Mandatory: no  # Range: 0-1024  # Default:  # LogFileSize=1    ### Option: DebugLevel  # Specifies debug level  # 0 - no debug  # 1 - critical information  # 2 - error information  # 3 - warnings  # 4 - for debugging (produces lots of information)  #  # Mandatory: no  # Range: 0-4  # Default:  # DebugLevel=3    ### Option: SourceIP  # Source IP address for outgoing connections.  #  # Mandatory: no  # Default:  # SourceIP=    ### Option: EnableRemoteCommands  # Whether remote commands from Zabbix server are allowed.  # 0 - not allowed  # 1 - allowed  #  # Mandatory: no  # Default:  # EnableRemoteCommands=0    ### Option: LogRemoteCommands  # Enable logging of executed shell commands as warnings.  # 0 - disabled  # 1 - enabled  #  # Mandatory: no  # Default:  # LogRemoteCommands=0    ##### Passive checks related    ### Option: Server  # List of comma delimited IP addresses (or hostnames) of Zabbix servers.  # Incoming connections will be accepted only from the hosts listed here.  # No spaces allowed.  # If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally.  #  # Mandatory: no  # Default:  # Server=zabbix-server-external.autoclouds.net    Server={{ zabbix\_server\_ip }}    ### Option: ListenPort  # Agent will listen on this port for connections from the server.  #  # Mandatory: no  # Range: 1024-32767  # Default:  ListenPort={{ zabbix\_port }}    ### Option: ListenIP  # List of comma delimited IP addresses that the agent should listen on.  # First IP address is sent to Zabbix server if connecting to it to retrieve list of active checks.  #  # Mandatory: no  # Default:  # ListenIP=0.0.0.0    ### Option: StartAgents  # Number of pre-forked instances of zabbix\_agentd that process passive checks.  # If set to 0, disables passive checks and the agent will not listen on any TCP port.  #  # Mandatory: no  # Range: 0-100  # Default:  # StartAgents=3    ##### Active checks related    ### Option: ServerActive  # List of comma delimited IP:port (or hostname:port) pairs of Zabbix servers for active checks.  # If port is not specified, default port is used.  # IPv6 addresses must be enclosed in square brackets if port for that host is specified.  # If port is not specified, square brackets for IPv6 addresses are optional.  # If this parameter is not specified, active checks are disabled.  # Example: ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,::1,[12fc::1]  #  # Mandatory: no  # Default:  # ServerActive=      ### Option: Hostname  # Unique, case sensitive hostname.  # Required for active checks and must match hostname as configured on the server.  # Value is acquired from HostnameItem if undefined.  #  # Mandatory: no  # Default:  # Hostname=    Hostname={{ ansible\_hostname }}    ### Option: HostnameItem  # Item used for generating Hostname if it is undefined.  # Ignored if Hostname is defined.  #  # Mandatory: no  # Default:  # HostnameItem=system.hostname    ### Option: RefreshActiveChecks  # How often list of active checks is refreshed, in seconds.  #  # Mandatory: no  # Range: 60-3600  # Default:  # RefreshActiveChecks=120    ### Option: BufferSend  # Do not keep data longer than N seconds in buffer.  #  # Mandatory: no  # Range: 1-3600  # Default:  # BufferSend=5    ### Option: BufferSize  # Maximum number of values in a memory buffer. The agent will send  # all collected data to Zabbix Server or Proxy if the buffer is full.  #  # Mandatory: no  # Range: 2-65535  # Default:  # BufferSize=100    ### Option: MaxLinesPerSecond  # Maximum number of new lines the agent will send per second to Zabbix Server  # or Proxy processing 'log' and 'logrt' active checks.  # The provided value will be overridden by the parameter 'maxlines',  # provided in 'log' or 'logrt' item keys.  #  # Mandatory: no  # Range: 1-1000  # Default:  # MaxLinesPerSecond=100    ### Option: AllowRoot  # Allow the agent to run as 'root'. If disabled and the agent is started by 'root', the agent  # will try to switch to user 'zabbix' instead. Has no effect if started under a regular user.  # 0 - do not allow  # 1 - allow  #  # Mandatory: no  # Default:  # AllowRoot=0    ############ ADVANCED PARAMETERS #################    ### Option: Alias  # Sets an alias for parameter. It can be useful to substitute long and complex parameter name with a smaller and simpler one.  #  # Mandatory: no  # Range:  # Default:    ### Option: Timeout  # Spend no more than Timeout seconds on processing  #  # Mandatory: no  # Range: 1-30  # Default:  Timeout=20    ### Option: Include  # You may include individual files or all files in a directory in the configuration file.  # Installing Zabbix will create include directory in /usr/local/etc, unless modified during the compile time.  #  # Mandatory: no  # Default:  # Include=    # Include=/usr/local/etc/zabbix\_agentd.userparams.conf  # Include=/usr/local/etc/zabbix\_agentd.conf.d/    ####### USER-DEFINED MONITORED PARAMETERS #######    ### Option: UnsafeUserParameters  # Allow all characters to be passed in arguments to user-defined parameters.  # 0 - do not allow  # 1 - allow  #  # Mandatory: no  # Range: 0-1  # Default:  # UnsafeUserParameters=0    ### Option: UserParameter  # User-defined parameter to monitor. There can be several user-defined parameters.  # Format: UserParameter=<key>,<shell command>  # See 'zabbix\_agentd' directory for examples.  #  # Mandatory: no  # Default:  # UserParameter=  UserParameter=memcached\_stats[\*],(echo stats; sleep 1) | telnet {{ ansible\_default\_ipv4.address }} $1 2>&1 | awk '/STAT $2 / {print $NF}'  UserParameter=mysql[\*],mysql -h {{ ansible\_default\_ipv4.address }} -P 3306 -uzabbix -pzabbix -e "show global status"|grep "\<$1\>"|cut -f2  UserParameter=redis\_stats[\*],(echo info; sleep 1) | telnet {{ ansible\_default\_ipv4.address }} $1 2>&1 |grep $2|cut -d : -f2  UserParameter=custom.vfs.dev.read.ops[\*],cat /proc/diskstats | grep $1 | head -1 | awk '{print $$4}'  UserParameter=custom.vfs.dev.read.ms[\*],cat /proc/diskstats | grep $1 | head -1 | awk '{print $$7}'  UserParameter=custom.vfs.dev.write.ops[\*],cat /proc/diskstats | grep $1 | head -1 | awk '{print $$8}'  UserParameter=custom.vfs.dev.write.ms[\*],cat /proc/diskstats | grep $1 | head -1 | awk '{print $$11}'  UserParameter=custom.vfs.dev.io.active[\*],cat /proc/diskstats | grep $1 | head -1 | awk '{print $$12}'  UserParameter=custom.vfs.dev.io.ms[\*],cat /proc/diskstats | grep $1 | head -1 | awk '{print $$13}'  UserParameter=custom.vfs.dev.read.sectors[\*],cat /proc/diskstats | grep $1 | head -1 | awk '{print $$6}'  UserParameter=custom.vfs.dev.write.sectors[\*],cat /proc/diskstats | grep $1 | head -1 | awk '{print $$10}'  UserParameter=MongoDB.Status[\*],/bin/echo "db.serverStatus().$1" | /usr/bin/mongo admin | grep "$2"|awk -F: '{print $$2}'|awk -F, '{print $$1}'  UserParameter=check\_lvm[\*],/usr/bin/sudo /usr/local/zabbix/bin/check\_lvm.sh $1  UserParameter=TCP\_ESTABLISHED,ss -s|awk 'NR==2{print $4}'|cut -d , -f1  UserParameter=TCP\_CLOSED,ss -s|awk 'NR==2{print $6}'|cut -d , -f1  UserParameter=TCP\_TIMEWAIT,ss -s|awk 'NR==2{print $12}'|cut -d / -f1  UserParameter=zabbix\_low\_discovery[\*],/bin/bash /usr/local/zabbix/bin/zabbix\_low\_discovery.sh $1  UserParameter=mysql\_stats[\*],mysql -h {{ ansible\_default\_ipv4.address }} -P $1 -uzabbix -pzabbix -e "show global status"|grep "\<$2\>"|cut -f2  UserParameter=mysql\_stats\_slave[\*],mysql -h {{ ansible\_default\_ipv4.address }} -P $1 -uzabbix -pzabbix -e "show slave status\G"|grep "\<$2\>"|awk '{if($NF=="Yes") {print 1} else {print 0}}'  UserParameter=check\_platform,dmidecode |grep Vendor|awk -F ' ' '{if($2=="Dell") {print 1} else {print 0}}'  #follow is hardware monitor  UserParameter=hardware\_battery,omreport chassis batteries|awk '/^Status/{if($NF=="Ok") {print 1} else {print 0}}'  UserParameter=hardware\_cpu\_model,awk -v hardware\_cpu\_crontol=`sudo omreport chassis biossetup|awk '/C State/{if($NF=="Enabled") {print 0} else {print 1}}'` -v hardware\_cpu\_c1=`sudo omreport chassis biossetup|awk '/C1[-|E]/{if($NF=="Enabled") {print 0} else {print 1}}'` 'BEGIN{if(hardware\_cpu\_crontol==0 && hardware\_cpu\_c1==0) {print 0} else {print 1}}'  UserParameter=hardware\_fan\_health,awk -v hardware\_fan\_number=`omreport chassis fans|grep -c "^Index"` -v hardware\_fan=`omreport chassis fans|awk '/^Status/{if($NF=="Ok") count+=1}END{print count}'` 'BEGIN{if(hardware\_fan\_number==hardware\_fan) {print 1} else {print 0}}'  UserParameter=hardware\_memory\_health,awk -v hardware\_memory=`omreport chassis memory|awk '/^Health/{print $NF}'` 'BEGIN{if(hardware\_memory=="Ok") {print 1} else {print 0}}'  UserParameter=hardware\_nic\_health,awk -v hardware\_nic\_number=`omreport chassis nics |grep -c "Interface Name"` -v hardware\_nic=`omreport chassis nics |awk '/^Connection Status/{print $NF}'|wc -l` 'BEGIN{if(hardware\_nic\_number==hardware\_nic) {print 1} else {print 0}}'  UserParameter=hardware\_cpu,omreport chassis processors|awk '/^Health/{if($NF=="Ok") {print 1} else {print 0}}'  UserParameter=hardware\_power\_health,awk -v hardware\_power\_number=`omreport chassis pwrsupplies|grep -c "Index"` -v hardware\_power=`omreport chassis pwrsupplies|awk '/^Status/{if($NF=="Ok") count+=1}END{print count}'` 'BEGIN{if(hardware\_power\_number==hardware\_power) {print 1} else {print 0}}'  UserParameter=hardware\_temp,omreport chassis temps|awk '/^Status/{if($NF=="Ok") {print 1} else {print 0}}'|head -n 1  UserParameter=hardware\_physics\_health,awk -v hardware\_physics\_disk\_number=`omreport storage pdisk controller=0|grep -c "^ID"` -v hardware\_physics\_disk=`omreport storage pdisk controller=0|awk '/^State/{if($NF=="Online") count+=1}END{print count}'` 'BEGIN{if(hardware\_physics\_disk\_number==hardware\_physics\_disk) {print 1} else {print 0}}'  UserParameter=hardware\_virtual\_health,awk -v hardware\_virtual\_disk\_number=`omreport storage vdisk controller=0|grep -c "^ID"` -v hardware\_virtual\_disk=`omreport storage vdisk controller=0|awk '/^State/{if($NF=="Ready") count+=1}END{print count}'` 'BEGIN{if(hardware\_virtual\_disk\_number==hardware\_virtual\_disk) {print 1} else {print 0}}'  UserParameter=pyora[\*],/usr/local/zabbix/bin/pyora.py --username $1 --password $2 --address $3 --database $4 $5 $6 $7 $8 |

此文件是客户端的配置文件

8、vars的main.yml

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 15:17:06 # cat /etc/ansible/roles/zabbix\_client\_install/vars/main.yml  zabbix\_dir: /usr/local 客户端安全目录  zabbix\_version: 2.0.6 客户端软件版本  zabbix\_user: zabbix 客户端用户  zabbix\_port: 10050 客户端的端口  zabbix\_server\_ip: 192.168.1.10 zabbix\_server的ip |

此文件是配置变量的

9、ansible安装zabbix客户端的playbook配置文件zabbix\_client\_install.yml

|  |
| --- |
| 15:20:02 # cat /etc/ansible/zabbix\_client\_install.yml  ---  - hosts: "{{host}}"  remote\_user: "{{user}}"  gather\_facts: True  roles:  - zabbix\_client\_install |

10、使用playbook安装zabbix客户端

我的测试客户端环境是centos 6.3，ip是192.168.240.17，使用key登陆

|  |
| --- |
| 15:22:01 # cd /etc/ansible/  root@ip-10-10-10-10:/etc/ansible  15:22:04 # time ansible-playbook zabbix\_client\_install.yml --extra-vars "host=192.168.240.17 user=root" --private-key=/root/test.pem    PLAY [192.168.240.17] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    GATHERING FACTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ok: [192.168.240.17]    TASK: [zabbix\_client\_install | Stop Exist Zabbix Client Service In Redhat Client] \*\*\*  failed: [192.168.240.17] => {"changed": true, "cmd": "ps -ef|grep zabbix|grep -v grep|awk '{print $2}'|xargs kill -9 >>/dev/null 2>&1 ", "delta": "0:00:00.018213", "end": "2014-07-10 07:22:34.793910", "item": "", "rc": 123, "start": "2014-07-10 07:22:34.775697"}  ...ignoring    TASK: [zabbix\_client\_install | Delete Exist Zabbix Client Dir In Redhat Client] \*\*\*  changed: [192.168.240.17]    TASK: [zabbix\_client\_install | Install Base Require Software In Redhat Client] \*\*\*  ok: [192.168.240.17] => (item=telnet,dmidecode,tar)    TASK: [zabbix\_client\_install | Create Zabbix User In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [zabbix\_client\_install | Copy Zabbix Client Software To Redhat Client] \*\*\*  changed: [192.168.240.17]    TASK: [zabbix\_client\_install | Uncompression Zabbix Client Software To Redhat Client] \*\*\*  changed: [192.168.240.17]    TASK: [zabbix\_client\_install | Copy Zabbix Start Script To Redhat Client] \*\*\*\*\*  changed: [192.168.240.17]    TASK: [zabbix\_client\_install | Copy Zabbix Config To Redhat Client] \*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [zabbix\_client\_install | Modify Zabbix Dir Permission In Redhat Client] \*\*\*  ok: [192.168.240.17]    TASK: [zabbix\_client\_install | Check Zabbix User Sudo Permission In Redhat Client] \*\*\*  changed: [192.168.240.17]    TASK: [zabbix\_client\_install | Give Sudo Permission To Zabbix User In Redhat Client] \*\*\*  changed: [192.168.240.17]    TASK: [zabbix\_client\_install | Start Zabbix Service In Redhat Client] \*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [zabbix\_client\_install | Add Boot Start Zabbix Service In Redhat Client] \*\*\*  changed: [192.168.240.17]    TASK: [zabbix\_client\_install | Delete Zabbix compression Software In Redhat Client] \*\*\*  changed: [192.168.240.17]    PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  192.168.240.17 : ok=15 changed=12 unreachable=0 failed=0      real 0m39.888s  user 0m1.547s  sys 0m0.197s |

可以看到39秒就安装完成，主要花费时间比较长的地方是fact收集、yum安装、文件传输。

11、测试安装情况

|  |
| --- |
| [root@ip-10-10-240-21 tmp]# ifconfig  eth0 Link encap:Ethernet HWaddr FA:16:3E:34:62:D0  inet addr:10.10.240.21 Bcast:10.10.240.255 Mask:255.255.255.0  inet6 addr: fe80::f816:3eff:fe34:62d0/64 Scope:Link  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  RX packets:542391 errors:0 dropped:0 overruns:0 frame:0  TX packets:77391 errors:0 dropped:0 overruns:0 carrier:0  collisions:0 txqueuelen:1000  RX bytes:142341119 (135.7 MiB) TX bytes:6451154 (6.1 MiB)    lo Link encap:Local Loopback  inet addr:127.0.0.1 Mask:255.0.0.0  inet6 addr: ::1/128 Scope:Host  UP LOOPBACK RUNNING MTU:16436 Metric:1  RX packets:10 errors:0 dropped:0 overruns:0 frame:0  TX packets:10 errors:0 dropped:0 overruns:0 carrier:0  collisions:0 txqueuelen:0  RX bytes:700 (700.0 b) TX bytes:700 (700.0 b)    [root@ip-10-10-240-21 tmp]# ps -ef|grep zabbix  zabbix 26991 1 0 07:22 ? 00:00:00 /usr/local/zabbix/sbin/zabbix\_agentd -c /usr/local/zabbix/conf/zabbix\_agentd.conf  zabbix 26993 26991 0 07:22 ? 00:00:00 /usr/local/zabbix/sbin/zabbix\_agentd -c /usr/local/zabbix/conf/zabbix\_agentd.conf  zabbix 26994 26991 0 07:22 ? 00:00:00 /usr/local/zabbix/sbin/zabbix\_agentd -c /usr/local/zabbix/conf/zabbix\_agentd.conf  zabbix 26995 26991 0 07:22 ? 00:00:00 /usr/local/zabbix/sbin/zabbix\_agentd -c /usr/local/zabbix/conf/zabbix\_agentd.conf  zabbix 26996 26991 0 07:22 ? 00:00:00 /usr/local/zabbix/sbin/zabbix\_agentd -c /usr/local/zabbix/conf/zabbix\_agentd.conf  root 27102 13773 0 07:24 pts/0 00:00:00 grep zabbix  [root@ip-10-10-240-21 tmp]# grep -Ev '^$|^#' /usr/local/zabbix/conf/zabbix\_agentd.conf  LogFile=/tmp/zabbix\_agentd.log  Server=192.168.1.10  ListenPort=10050  Hostname=ip-10-10-240-21  Timeout=20  UserParameter=memcached\_stats[\*],(echo stats; sleep 1) | telnet 10.10.240.21 $1 2>&1 | awk '/STAT $2 / {print $NF}'  UserParameter=mysql[\*],mysql -h 10.10.240.21 -P 3306 -uzabbix -pzabbix -e "show global status"|grep "\<$1\>"|cut -f2  UserParameter=redis\_stats[\*],(echo info; sleep 1) | telnet 10.10.240.21 $1 2>&1 |grep $2|cut -d : -f2  UserParameter=custom.vfs.dev.read.ops[\*],cat /proc/diskstats | grep $1 | head -1 | awk '{print $$4}'  UserParameter=custom.vfs.dev.read.ms[\*],cat /proc/diskstats | grep $1 | head -1 | awk '{print $$7}'  UserParameter=custom.vfs.dev.write.ops[\*],cat /proc/diskstats | grep $1 | head -1 | awk '{print $$8}'  UserParameter=custom.vfs.dev.write.ms[\*],cat /proc/diskstats | grep $1 | head -1 | awk '{print $$11}'  UserParameter=custom.vfs.dev.io.active[\*],cat /proc/diskstats | grep $1 | head -1 | awk '{print $$12}'  UserParameter=custom.vfs.dev.io.ms[\*],cat /proc/diskstats | grep $1 | head -1 | awk '{print $$13}'  UserParameter=custom.vfs.dev.read.sectors[\*],cat /proc/diskstats | grep $1 | head -1 | awk '{print $$6}'  UserParameter=custom.vfs.dev.write.sectors[\*],cat /proc/diskstats | grep $1 | head -1 | awk '{print $$10}'  UserParameter=MongoDB.Status[\*],/bin/echo "db.serverStatus().$1" | /usr/bin/mongo admin | grep "$2"|awk -F: '{print $$2}'|awk -F, '{print $$1}'  UserParameter=check\_lvm[\*],/usr/bin/sudo /usr/local/zabbix/bin/check\_lvm.sh $1  UserParameter=TCP\_ESTABLISHED,ss -s|awk 'NR==2{print $4}'|cut -d , -f1  UserParameter=TCP\_CLOSED,ss -s|awk 'NR==2{print $6}'|cut -d , -f1  UserParameter=TCP\_TIMEWAIT,ss -s|awk 'NR==2{print $12}'|cut -d / -f1  UserParameter=zabbix\_low\_discovery[\*],/bin/bash /usr/local/zabbix/bin/zabbix\_low\_discovery.sh $1  UserParameter=mysql\_stats[\*],mysql -h 10.10.240.21 -P $1 -uzabbix -pzabbix -e "show global status"|grep "\<$2\>"|cut -f2  UserParameter=mysql\_stats\_slave[\*],mysql -h 10.10.240.21 -P $1 -uzabbix -pzabbix -e "show slave status\G"|grep "\<$2\>"|awk '{if($NF=="Yes") {print 1} else {print 0}}'  UserParameter=check\_platform,dmidecode |grep Vendor|awk -F ' ' '{if($2=="Dell") {print 1} else {print 0}}'  UserParameter=hardware\_battery,omreport chassis batteries|awk '/^Status/{if($NF=="Ok") {print 1} else {print 0}}'  UserParameter=hardware\_cpu\_model,awk -v hardware\_cpu\_crontol=`sudo omreport chassis biossetup|awk '/C State/{if($NF=="Enabled") {print 0} else {print 1}}'` -v hardware\_cpu\_c1=`sudo omreport chassis biossetup|awk '/C1[-|E]/{if($NF=="Enabled") {print 0} else {print 1}}'` 'BEGIN{if(hardware\_cpu\_crontol==0 && hardware\_cpu\_c1==0) {print 0} else {print 1}}'  UserParameter=hardware\_fan\_health,awk -v hardware\_fan\_number=`omreport chassis fans|grep -c "^Index"` -v hardware\_fan=`omreport chassis fans|awk '/^Status/{if($NF=="Ok") count+=1}END{print count}'` 'BEGIN{if(hardware\_fan\_number==hardware\_fan) {print 1} else {print 0}}'  UserParameter=hardware\_memory\_health,awk -v hardware\_memory=`omreport chassis memory|awk '/^Health/{print $NF}'` 'BEGIN{if(hardware\_memory=="Ok") {print 1} else {print 0}}'  UserParameter=hardware\_nic\_health,awk -v hardware\_nic\_number=`omreport chassis nics |grep -c "Interface Name"` -v hardware\_nic=`omreport chassis nics |awk '/^Connection Status/{print $NF}'|wc -l` 'BEGIN{if(hardware\_nic\_number==hardware\_nic) {print 1} else {print 0}}'  UserParameter=hardware\_cpu,omreport chassis processors|awk '/^Health/{if($NF=="Ok") {print 1} else {print 0}}'  UserParameter=hardware\_power\_health,awk -v hardware\_power\_number=`omreport chassis pwrsupplies|grep -c "Index"` -v hardware\_power=`omreport chassis pwrsupplies|awk '/^Status/{if($NF=="Ok") count+=1}END{print count}'` 'BEGIN{if(hardware\_power\_number==hardware\_power) {print 1} else {print 0}}'  UserParameter=hardware\_temp,omreport chassis temps|awk '/^Status/{if($NF=="Ok") {print 1} else {print 0}}'|head -n 1  UserParameter=hardware\_physics\_health,awk -v hardware\_physics\_disk\_number=`omreport storage pdisk controller=0|grep -c "^ID"` -v hardware\_physics\_disk=`omreport storage pdisk controller=0|awk '/^State/{if($NF=="Online") count+=1}END{print count}'` 'BEGIN{if(hardware\_physics\_disk\_number==hardware\_physics\_disk) {print 1} else {print 0}}'  UserParameter=hardware\_virtual\_health,awk -v hardware\_virtual\_disk\_number=`omreport storage vdisk controller=0|grep -c "^ID"` -v hardware\_virtual\_disk=`omreport storage vdisk controller=0|awk '/^State/{if($NF=="Ready") count+=1}END{print count}'` 'BEGIN{if(hardware\_virtual\_disk\_number==hardware\_virtual\_disk) {print 1} else {print 0}}'  UserParameter=pyora[\*],/usr/local/zabbix/bin/pyora.py --username $1 --password $2 --address $3 --database $4 $5 $6 $7 $8  [root@ip-10-10-240-21 tmp]# ll /tmp/  total 12  -rw------- 1 root root 197 Jul 9 09:35 yum\_save\_tx-2014-07-09-09-35ibcBiO.yumtx  -rw-rw-r-- 1 zabbix zabbix 320 Jul 10 07:22 zabbix\_agentd.log  -rw-rw-r-- 1 zabbix zabbix 5 Jul 10 07:22 zabbix\_agentd.pid  [root@ip-10-10-240-21 tmp]# chkconfig --list|grep zabbix\_agentd  zabbix\_agentd 0:off 1:off 2:off 3:on 4:on 5:on 6:off  [root@ip-10-10-240-21 tmp]# grep zabbix /etc/sudoers  zabbix ALL=(root) NOPASSWD:/bin/netstat, /usr/bin/omreport  [root@ip-10-10-240-21 tmp]# ll /etc/init.d/zabbix\_agentd  -rwxr-xr-x 1 root root 1444 Jul 10 07:22 /etc/init.d/zabbix\_agentd |

可以看到安装后的客户端，完全是按照我的要求来做的。

12、删除已经安装的客户端

|  |
| --- |
| 15:22:54 # time ansible-playbook zabbix\_client\_delete.yml --extra-vars "host=192.168.240.17 user=root" --private-key=/root/test.pem    PLAY [192.168.240.17] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    GATHERING FACTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ok: [192.168.240.17]    TASK: [zabbix\_client\_delete | Stop Zabbix Service In RedHat Client] \*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [zabbix\_client\_delete | Delete Boot Start Zabbix Service In Redhat Client] \*\*\*  changed: [192.168.240.17]    TASK: [zabbix\_client\_delete | Delete Zabbix User In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [zabbix\_client\_delete | Delete Zabbix Dir In Redhat Client] \*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [192.168.240.17]    TASK: [zabbix\_client\_delete | Delete Zabbix Start Script In Redhat Client] \*\*\*\*  changed: [192.168.240.17]    TASK: [zabbix\_client\_delete | Check Zabbix User Sudo Permission In Redhat Client] \*\*\*  changed: [192.168.240.17]    TASK: [zabbix\_client\_delete | Delete Sudo Permission To Zabbix User In Redhat Client] \*\*\*  changed: [192.168.240.17]    PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  192.168.240.17 : ok=8 changed=7 unreachable=0 failed=0      real 0m25.497s  user 0m0.847s  sys 0m0.159s |

13、测试删除情况

|  |
| --- |
| [root@ip-10-10-240-21 tmp]# ll /tmp/  total 4  -rw------- 1 root root 197 Jul 9 09:35 yum\_save\_tx-2014-07-09-09-35ibcBiO.yumtx  [root@ip-10-10-240-21 tmp]# ps -ef|grep zabbix  root 27665 13773 0 07:27 pts/0 00:00:00 grep zabbix  [root@ip-10-10-240-21 tmp]# ll /usr/local/  total 40  drwxr-xr-x. 2 root root 4096 Sep 23 2011 bin  drwxr-xr-x. 2 root root 4096 Sep 23 2011 etc  drwxr-xr-x. 2 root root 4096 Sep 23 2011 games  drwxr-xr-x. 2 root root 4096 Sep 23 2011 include  drwxr-xr-x. 2 root root 4096 Sep 23 2011 lib  drwxr-xr-x. 2 root root 4096 Sep 23 2011 lib64  drwxr-xr-x. 2 root root 4096 Sep 23 2011 libexec  drwxr-xr-x. 2 root root 4096 Sep 23 2011 sbin  drwxr-xr-x. 5 root root 4096 May 12 2013 share  drwxr-xr-x. 3 root root 4096 May 13 2013 src  [root@ip-10-10-240-21 tmp]# grep zabbix /etc/sudoers  [root@ip-10-10-240-21 tmp]# ll /etc/init.d/zabbix\_agentd  ls: cannot access /etc/init.d/zabbix\_agentd: No such file or directory  [root@ip-10-10-240-21 tmp]# chkconfig --list|grep zabbix\_agentd  [root@ip-10-10-240-21 tmp]# |

可以看到已经完全删除。

如果大家想使用我的例子，可以从附件里下载，然后放到/etc/ansible目录里，下面是压缩包里的内容

|  |
| --- |
| -rw-r--r-- root/root 108 2014-07-10 15:20 zabbix\_client\_install.yml  -rw-r--r-- root/root 121 2014-07-09 18:09 zabbix\_client\_delete.yml  drwxr-xr-x root/root 0 2014-07-01 16:38 roles/zabbix\_client\_install/  drwxr-xr-x root/root 0 2014-07-08 14:30 roles/zabbix\_client\_install/meta/  -rw-r--r-- root/root 207 2014-07-08 14:30 roles/zabbix\_client\_install/meta/main.yml  drwxr-xr-x root/root 0 2014-07-10 14:07 roles/zabbix\_client\_install/tasks/  -rw-r--r-- root/root 199 2014-07-10 14:02 roles/zabbix\_client\_install/tasks/delete.yml  -rw-r--r-- root/root 65 2014-07-10 14:02 roles/zabbix\_client\_install/tasks/main.yml  -rw-r--r-- root/root 1789 2014-07-10 14:02 roles/zabbix\_client\_install/tasks/copy.yml  -rw-r--r-- root/root 1110 2014-07-10 14:07 roles/zabbix\_client\_install/tasks/install.yml  drwxr-xr-x root/root 0 2014-06-19 13:30 roles/zabbix\_client\_install/handlers/  drwxr-xr-x root/root 0 2014-07-09 17:54 roles/zabbix\_client\_install/vars/  -rw-r--r-- root/root 115 2014-07-09 17:54 roles/zabbix\_client\_install/vars/main.yml  drwxr-xr-x root/root 0 2014-07-09 17:53 roles/zabbix\_client\_install/templates/  -rw-r--r-- zabbix/zabbix 10465 2014-07-09 17:53 roles/zabbix\_client\_install/templates/zabbix\_agentd.conf  -rwxr-xr-x root/root 1456 2014-07-08 15:20 roles/zabbix\_client\_install/templates/zabbix\_agentd  drwxr-xr-x root/root 0 2014-07-09 17:13 roles/zabbix\_client\_install/files/  -rw-r--r-- root/root 292293 2014-07-09 17:13 roles/zabbix\_client\_install/files/zabbix-2.0.6.tar.gz  drwxr-xr-x root/root 0 2014-06-23 14:03 roles/zabbix\_client\_delete/  drwxr-xr-x root/root 0 2014-07-09 18:08 roles/zabbix\_client\_delete/meta/  -rw-r--r-- root/root 205 2014-07-09 18:08 roles/zabbix\_client\_delete/meta/main.yml  drwxr-xr-x root/root 0 2014-07-10 14:28 roles/zabbix\_client\_delete/tasks/  -rw-r--r-- root/root 1518 2014-07-10 14:08 roles/zabbix\_client\_delete/tasks/delete.yml  -rw-r--r-- root/root 22 2014-07-10 14:08 roles/zabbix\_client\_delete/tasks/main.yml  drwxr-xr-x root/root 0 2014-06-24 14:14 roles/zabbix\_client\_delete/handlers/  drwxr-xr-x root/root 0 2014-07-03 13:16 roles/zabbix\_client\_delete/vars/  -rw-r--r-- root/root 115 2014-07-09 17:55 roles/zabbix\_client\_delete/vars/main.yml  drwxr-xr-x root/root 0 2014-07-09 18:08 roles/zabbix\_client\_delete/templates/  drwxr-xr-x root/root 0 2014-06-24 13:53 roles/zabbix\_client\_delete/files/ |

后续我会继续介绍使用playbook安装其他软件的例子。

如果觉得好，给个赞并多评论，谢谢。