# 安装ansible

http://sofar.blog.51cto.com/353572/1579894

1.安装setuptools

wget http://pypi.python.org/packages/source/s/setuptools/setuptools-0.6c11.tar.gz  
tar zxvf setuptools-0.6c11.tar.gz  
cd setuptools-0.6c11  
python setup.py build  
python setup.py install

安装完setuptools这个模块，再回到pip-1.5.4目录执行python setup.py install以便安装pip

2.安装pip

wget "https://pypi.python.org/packages/source/p/pip/pip-1.5.4.tar.gz#md5=834b2904f92d46aaa333267fb1c922bb" --no-check-certificate

# tar -xzvf pip-1.5.4.tar.gz

# cd pip-1.5.4p-1.5.4

# python setup.py install

3.安装好pip，再执行sudo pip install ansible即可安装ansible



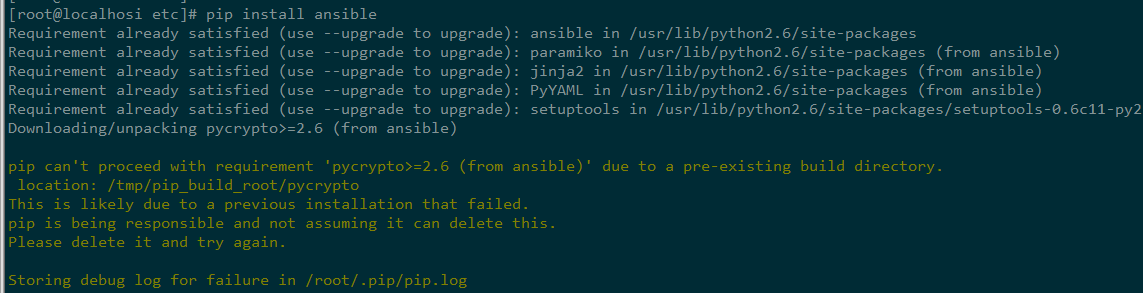
安装过程如果报如下错误，原因是少了python的dev



解决方法：yum install python-devel

报如下错误，是说/tmp/pip\_build\_root/目录下已存在pycrypto文件，可能是之前安装错误时遗留下来的，请删除该文件再重新安装

解决方法：删除/tmp/pip\_build\_root/目录下的pycrypto文件，再pip install ansible安装



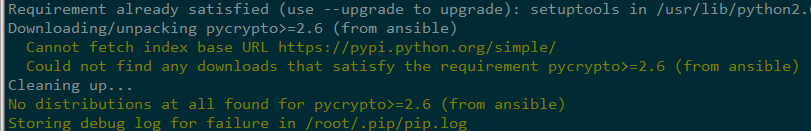
出现下面错误，说明python没有安装pycrypto组件

解决方法：<https://pypi.python.org/pypi/pycrypto> 下载pycrypto

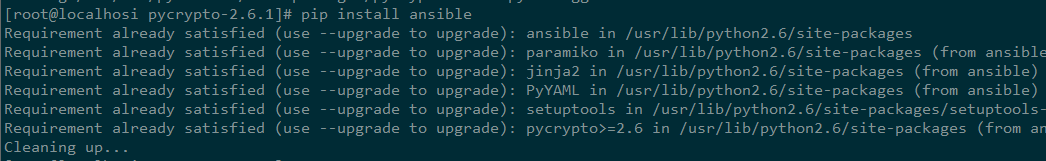
然后使用命令就可以安装成功了：

python setup.py build

python setup.py install

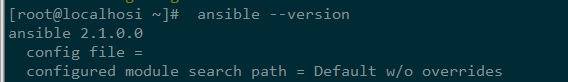


再次使用pip install ansible安装即可

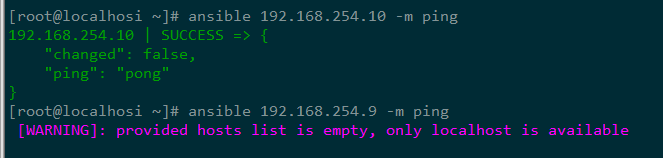


#### 测试ansible是否安装ok

 ansible --version



ansible -i hosts all -m ping –k    # 输入密码测试成功安装OK



默认安装好ansible是没有ansible目录的，需要手动创建

Mkdir /etc/ansible

Touch hosts



编译hosts文件

内容为INI格式：

192.168.1.134

192.168.1.136

也可以进行分组，同一个机器可以在不同分组

[v1]

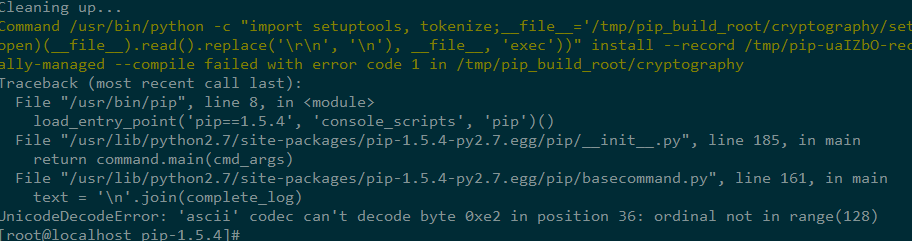
192.168.1.134

[v2]

192.168.1.136

如果远程的ssh端口修改了不在22标准端口，文件内容中机器地址应该明确指定端口号如：192.168.1.134:2222

#### 报错：

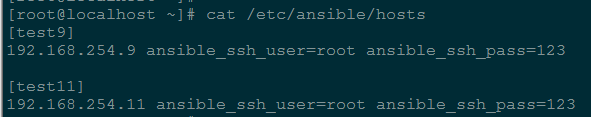


# 完整安装ansible

http://www.linuxidc.com/Linux/2015-02/112774.htm

# ansible命令

#### 免密码登录



直接在hosts文件写上账号和密码

配置ssh免密码登录远程主机：

# 当前用户名与远程机器上相同，ssh key在默认位置

ssh-copy-id 192.168.1.136

ssh-copy-id 192.168.1.134

避免重复输入密钥的短语密码：

ssh-add ~/.ssh/id\_rsa

在本地测试是否能够ping通：

# ping全部机器

ansible all -m ping

# 以ashin用户身份ping .134

ansible 192.168.1.134 -m ping -u ashin

# 以用户ashin身份使用sudo来ping v1 (.134)

# -K是输入root密码

ansible v1 -m ping -u ashin --sudo -K

**配置文件**

配置文件可以从多个地方加载，其优先级顺序为：

* ANSIBLE\_CONFIG (环境变量)
* ansible.cfg (当前目录)
* .ansible.cfg (home目录)
* /etc/ansible/ansible.cfg

定义自己的配置文件可以参考<https://raw.githubusercontent.com/ansible/ansible/devel/examples/ansible.cfg>

ssh检查key是通过paramiko进行的，很慢。 不检查host key配置：

sudo vi etc/ansible/ansible.cfg

# or ~/.ansible.cfg

内容为：

[defaults]

host\_key\_checking = False

#### 执行Ad-Hoc命令(就是无需保存的快速执行shell命令)：

使用-a指定命令， -f指定并发数（默认为5）, -m是选择使用的模块：

**执行shell**

# 打印hello

ansible all -a "/bin/echo hello"

# 重启

ansible all -a "/sbin/reboot" -f 10 --sudo -K

# 使用shell模块 注意当前shell的引号问题

ansible all -m shell -a 'ls -al ~'

**文件操作**

# 传输文件

ansible all -m copy -a "src=~/projects/tests/t.py dest=~"

# 修改文件权限，所有者，分组（这些参数可以用在copy模块中）

ansible all -m file -a "dest=~/t.py mode=777 owner=ashin group=ashin"

# 创建文件夹

ansible all -m file -a "dest=~/tests mode=755 owner=ashin group=ashin state=directory"

# 删除文件夹

ansible all -m file -a "dest=~/tests state=absent"

**安装软件**

使用yum或apt模块可以进行软件的安装

# 确保某个程序已经安装，并保持当前版本，如果没安装则进行安装

ansible v1 -m apt -a "name=python-pip state=present" --sudo -K

# 确保安装最新版本

ansible v1 -m apt -a "name=git state=latest"

# 确保没有安装某个程序，安装了则卸载

ansible v1 -m apt -a "name=git state=absent" --sudo -K

**用户管理**

# 新建用户

ansible all -m user -a "name=foo password=foo" --sudo -K

# 删除用户

ansible all -m user -a "name=foo state=absent" --sudo -K

**Git**

# 使用https方式检出代码到本地，前提是要先通过apt模块安装git

# dest目录必须是空文件夹或者还不存在的文件夹

ansible v1 -m git -a "repo=https://github.com/axiaoxin/json\_line\_format.vim.git dest=~/project-dir version=HEAD"

**服务管理**

# 确保某服务已开启，没开则开

ansible v1 -m service -a "name=mysql state=started" --sudo -K

ansible v1 -m service -a "name=mysql state=restarted" --sudo -K

ansible v1 -m service -a "name=mysql state=stopped" --sudo -K

ad-hoc常用命令是用来解决一些常用的简单的配置命令，而且这些命令的执行速度很快。adhoc可以不用写playbooks，使用方便简单。

Ansible all -m ping 检查所有主机存活状态

Ansible local –m ping 检查local组主机存活状态

ansible-doc –l 查看ansible支持的模块

openssl passwd  -1 "12321421421" 生产加密的密码

-1 表示用MD5基于BSD的密钥算法

**ansible使用格式**

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

    host-pattern # 可以是all，或者配置文件中的主机组名

    -f forks  # 指定并行处理的进程数

    -m module # 指定使用的模块，默认模块为command

    -a args   # 指定模块的参数

如果你有多台服务器的话，想并发运行，可以使用-f参数，默认是并发5

默认ansible是使用key验证的，如果使用密码登陆的服务器，使用ansible的话，要不修改ansible.cfg配置文件的 ask\_pass      = True给取消注释，要不就在运行命令时候加上-k

**创建SSH公钥与私钥**

 [root@localhost ~]# ssh-keygen

**将公钥文件复制到目标服务器  [注: ssh-copy-id 把公钥追加到远程主机的 .ssh/authorized\_key 上.]**

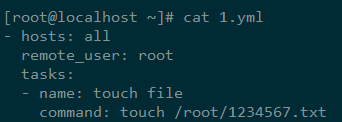
[root@localhost ~]# ssh-copy-id root@192.168.254.10

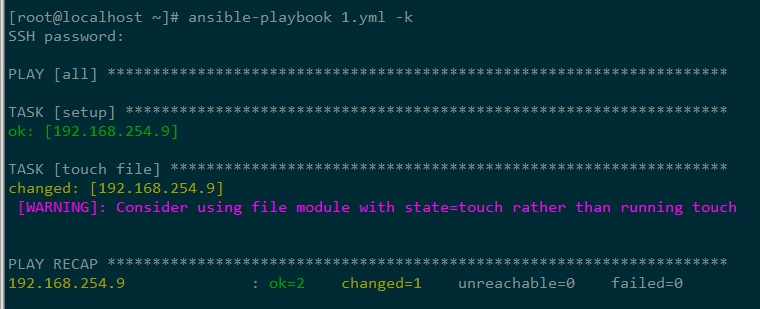
[root@localhost ~]# ssh-copy-id root@192.168.254.12

**连接与验证测试**

[root@localhost ~]# ansible -i /etc/ansible/hosts all -m ping

# Playbooks





由target（目标）、variable（变量）、task（任务）和handler（处理者）组成

Target定义将要执行playbook的远程主机组

Variable定义playbook运行时需要使用的变量

Task定义将要在远程主机上执行的任务列表

Handler定义task执行完成以后需要调用的任务

#### Target section常用参数

1. hosts：定义远程主机组
2. user：执行该任务的用户
3. sudo: 设置为yes的时候，执行任务的时候使用root权限
4. sudo\_user 如果你设置用户为**tshare365**，那么你执行的时候会使用**tshare365**用户的权限
5. connection 通过什么方式连接到远程主机，默认是ssh
6. gather\_facks 是否启用在远程主机执行setup模块，默认是会执行的，可用同setup模块获取远程主机的信息，在定义变量的时候使用

#### Variabler section常用参数

1. vars  定义格式 变量名:变量值
2. vars\_files  指定变量文件
3. vars\_prompt  用户交互模式自定义变量
4. setup 模块去远程主机的值

#### Task ssection常用参数

1. name：输出到屏幕的信息
2. action：定义执行的动作调用ansible的模块例如：yum name=http state=installed就是安装apache服务
3. copy：复制本地文件到远程主机
4. template：复制本地文件到远程主机但是他可以在本地文件中调用变量
5. service ：定义服务的状态

## **YAML语法**

以下是YML语法的三个基本规则：

1、YML使用缩进来表示层级关系，缩进用2个空格，不能使用table

2、冒号后面有一个空格，除了表示路径的情况以外

      如：   my\_key: my\_value

3、短横线表示列表，短横线后面要有一个空格，如：

- list\_value\_one

- list\_value\_tow

- list\_value\_three

playbooks使用yaml语法，在ansible中几乎所有的yaml文件都是以list开始，每个item是键值对的list。

所有的yaml文件都以---开头表示开始一个document，所有的列表元素以-开头，键值对用:，后面的空格是必须的。（如果官网文档中有指出空格那一定是merge了我的pull request^\_^)

键值对中的值如果是bool类型的字符串true/false（首字母不论大小写），pyyaml会load成python中对应的bool值，在键值对中如果值中有:或者以{{开头的变量定义，则必须用引号引起来

**ping示例**

新建文件test.yml内容为：

**---**

- hosts: v1

remote\_user: ashin # 如果和当前用户一样，则无需指定

tasks: # tasks是是关键词，指明了要执行哪些任务

- name: test connection #name是任务的名称

shell: ‘ping [www.baidu](http://www.baidu)’ #shell是前面提到的module(模块)，单引号中是命令

运行：

ansible-playbook test.yml

一个playbook文件可以引入其他的yml文件，实现复用。- include: path/other.yml

# ansible.cfg与默认配置

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：

1. [defaults]
2. # some basic default values...
3. hostfile = /etc/ansible/hosts \\指定默认hosts配置的位置
4. # library\_path = /usr/share/my\_modules/
5. remote\_tmp = $HOME/.ansible/tmp
6. pattern = \*
7. forks = 5
8. poll\_interval = 15
9. sudo\_user = root \\远程sudo用户
10. #ask\_sudo\_pass = True \\每次执行ansible命令是否询问ssh密码
11. #ask\_pass = True \\每次执行ansible命令是否询问sudo密码
12. transport = smart
13. remote\_port = 22
14. module\_lang = C
15. gathering = implicit
16. host\_key\_checking = False \\关闭第一次使用ansible连接客户端是输入命令提示
17. log\_path = /var/log/ansible.log \\需要时可以自行添加。chown -R root:root ansible.log
18. system\_warnings = False \\关闭运行ansible时系统的提示信息，一般为提示升级
19. # set plugin path directories here, separate with colons
20. action\_plugins = /usr/share/ansible\_plugins/action\_plugins
21. callback\_plugins = /usr/share/ansible\_plugins/callback\_plugins
22. connection\_plugins = /usr/share/ansible\_plugins/connection\_plugins
23. lookup\_plugins = /usr/share/ansible\_plugins/lookup\_plugins
24. vars\_plugins = /usr/share/ansible\_plugins/vars\_plugins
25. filter\_plugins = /usr/share/ansible\_plugins/filter\_plugins
26. fact\_caching = memory
27. [accelerate]
28. accelerate\_port = 5099
29. accelerate\_timeout = 30
30. accelerate\_connect\_timeout = 5.0
31. # The daemon timeout is measured in minutes. This time is measured
32. # from the last activity to the accelerate daemon.
33. accelerate\_daemon\_timeout = 30

本篇就结合一个示例对其进行下了解。我在对之前未连接的主机进行连结时报错如下：

1. [root@361way.com ~]# ansible test -a 'uptime'
2. 10.212.52.14 | 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.
3. 10.212.52.16 | 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.

从上面的输出提示上基本可以了解到由于在本机的~/.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：

了解到问题原因为，我们了解到进行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：

1. [ssh\_connection]
2. # ssh arguments to use
3. # Leaving off ControlPersist will result in poor performance, so use
4. # paramiko on older platforms rather than removing it
5. #ssh\_args = -o ControlMaster=auto -o ControlPersist=60s

所以这里我们可以启用ssh\_args 部分，使用下面的配置，避免上面出现的错误：

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

方法2：

在ansible.cfg配置文件中，也会找到如下部分：

1. # uncomment this to disable SSH key host checking
2. 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。由于配置文件中直接有该选项，所以推荐用方法2 。

其他部分

由于官方给的说明比较详细，同时ansible.cfg 文件本身默认也有注释提示部分，所以不做过多说明，这里再举个例子，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：

1. # logging is off by default unless this path is defined
2. # if so defined, consider logrotate
3. log\_path = /var/log/ansible.log

同样，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，便于了解在何时执行了何操作及其结果，如下：

1. [root@361way.com ansible]# cat /var/log/ansible.log
2. 2015-05-04 01:57:19,758 p=4667 u=root |
3. 2015-05-04 01:57:19,759 p=4667 u=root | /usr/bin/ansible test -a uptime
4. 2015-05-04 01:57:19,759 p=4667 u=root |
5. 2015-05-04 01:57:20,563 p=4667 u=root | 10.212.52.252 | success | rc=0 >>
6. 01:57am up 23 days 11:20, 2 users, load average: 0.38, 0.38, 0.40
7. 2015-05-04 01:57:20,831 p=4667 u=root | 10.212.52.14 | success | rc=0 >>
8. 02:03am up 331 days 8:19, 2 users, load average: 0.08, 0.05, 0.05
9. 2015-05-04 01:57:20,909 p=4667 u=root | 10.212.52.16 | success | rc=0 >>
10. 02:05am up 331 days 8:56, 2 users, load average: 0.00, 0.01, 0.05

# 常用命令

## ****用户管理****

# 新建用户

ansible all -m user -a "name=foo password=foo" --sudo -K

# 删除用户

ansible all -m user -a "name=foo state=absent" --sudo -K

## ****执行shell****

# 打印hello

ansible all -a "/bin/echo hello"

# 重启

ansible all -a "/sbin/reboot" -f 10 --sudo -K

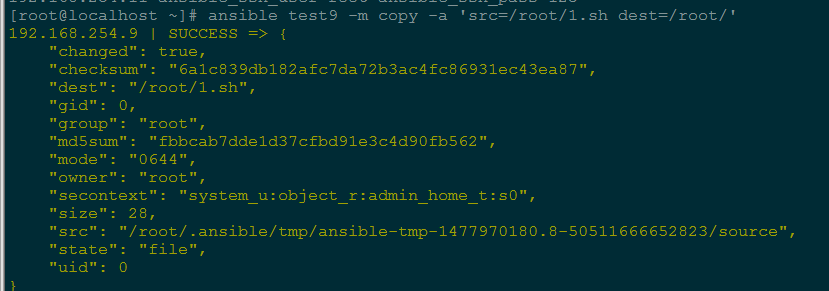
# 使用shell模块 注意当前shell的引号问题

ansible all -m shell -a 'ls -al ~'

## **文件操作**

# 传输文件

ansible all -m copy -a "src=~/projects/tests/t.py dest=~"



# 修改文件权限，所有者，分组（这些参数可以用在copy模块中）

ansible all -m file -a "dest=~/t.py mode=777 owner=ashin group=ashin"

# 创建文件夹

ansible all -m file -a "dest=~/tests mode=755 owner=ashin group=ashin state=directory"

# 删除文件夹

ansible all -m file -a "dest=~/tests state=absent"

## **安装软件：使用yum或apt模块可以进行软件的安装**

# 确保某个程序已经安装，并保持当前版本，如果没安装则进行安装

ansible v1 -m apt -a "name=python-pip state=present" --sudo -K

# 确保安装最新版本

ansible v1 -m apt -a "name=git state=latest"

# 确保没有安装某个程序，安装了则卸载

ansible v1 -m apt -a "name=git state=absent" --sudo -K

## **服务管理**

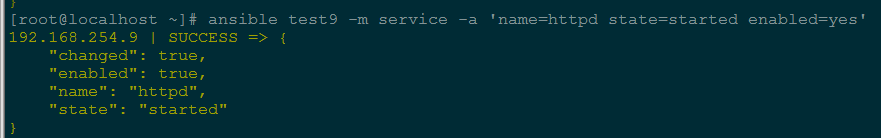
# 确保某服务已开启，没开则开

ansible v1 -m service -a "name=mysql state=started" --sudo -K

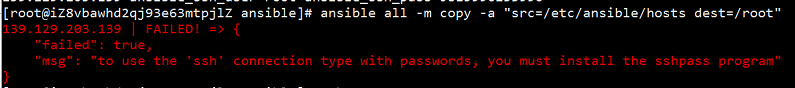
ansible v1 -m service -a "name=mysql state=restarted" --sudo -K

ansible v1 -m service -a "name=mysql state=stopped" --sudo -K

启动服务



报错：



安装sshpass yum install -y sshpass