- authorized_key
- file模块
- copy模块
- yum模块
- service模块
- systemd模块
- cron模块
- group模块
- user模块
- synchronize模块
- filesystem模块
- mount模块
- get_url 模块
- unarchive模块
- assemble

根据官方的分类，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。 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。

# 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
```

# authorized_key

用于向被控端推送公钥，通常用于在ansible第一次连接被控端时向其推送ansible主控端的管理公钥。

常用选项：

- user: 指定将公钥推送给被控端的哪个用户
- key：指定被推送的公钥的内容
- path：默认情况下，会将公钥推送至被控端用户家目录的.ssh/authorized_keys文件中，可通过该配置项自定义该路径
- state：是推送还是删除，present|absent

示例：

```
ansible test -m authorized_key -a 'user=ansible key="{{
lookup('file','/root/.ssh/id_rsa.pub')}}"'
```

> 更多用法可参考官方文档

# 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
- 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 192.168.8.120 -m file -a 'path=/tmp/test.txt state=touch owner=root group=root
mode=644'

ansible 192.168.8.120 -m file -a 'src=/tmp/test.txt dest=/root/test.txt state=link'

ansible 192.168.8.120 -m file -a 'path=/tmp/test.txt state=file'

ansible 192.168.8.120 -m file -a 'path=/tmp/test state=directory owner=root group=root
mode=755'

ansible 192.168.8.120 -m file -a 'path=/tmp/test2/test3/aaa/bbb state=directory
owner=root group=root mode=755'

ansible 192.168.8.120 -m file -a 'path=/tmp/test2 state=absent'
```

> 更多用法可参考官方文档

# 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
- 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模块里的相关文件属性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。

示例如下：

```
ansible 192.168.8.120 -m copy -a 'src=/etc/ansible/ansible.cfg dest=/usr/local/src/
owner=root group=root mode=644'

ansible 192.168.8.120 -m copy -a 'backup=yes src=/etc/fstab
dest=/usr/local/src/ansible.cfg owner=root group=root mode=644'

ansible 192.168.8.120 -m copy -a 'content="just a test!" dest=/usr/local/src/test.txt'

ansible 192.168.8.120 -m copy -a 'src=/data dest=/usr/local/src/'

ansible 192.168.8.120 -m copy -a 'src=/data/ dest=/usr/local/src/'

ansible 192.168.8.120 -m copy -a "src=/mine/sudoers dest=/etc/sudoers validate='visudo -
cf %s'"
```

# yum模块

使用yum包管理器来管理软件包，其选项有：

- `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'
```

# 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）
- daemon_reload：针对使用systemd的系统，重新加载systemd配置，yes/no

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"
```

# systemd模块

- daemon_reload：当服务配置文件发生变更时重载服务
- name：指定服务名称
- enabled：是否设置开机自启
- state：管理服务状态，reloaded|restarted|started|stopped

示例：

```
ansible test -m systemd -a "name=httpd state=started enabled=yes daemon_reload=yes"
```

# 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'
```

# group模块

goup模块请求的是groupadd, groupdel, groupmod 三个指令。

- gid：指定组id
- name：指定组名
- state：创建还是删除组，选项：present|absent
- system：是否将该组创建为系统组，默认为no

示例：

```
ansible test -m group -a 'name=test gid=1001 state=present system=yes'
```

# user模块

user模块是请求的是useradd, userdel, usermod三个指令

- home：指定用户的家目录，需要与createhome配合使用
- groups：指定用户的附加组
- group：指定用户属组
- 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
- comment：定义用户描述信息
- 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
#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
user: name=test generate_ssh_key=yes ssh_key_bits=2048 ssh_key_file=.ssh/id_rsa
```

需要说明的是，在指定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 7使用的是SHA512加密算法。

# 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：传输目录不进行递归，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模式的话，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
```

# filesystem模块

在块设备上创建文件系统
常用选项：

- 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"'
```

# mount模块

配置挂载点
选项：

- boot：是否开机自动挂载
- 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'
```

# 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
```

# unarchive模块

用于解压文件，模块包含如下选项：

- remote_src：如果为yes，则文件会从master端复制到目标端。否则会直接尝试从目标端查找文件。默认为no。
- creates：指定一个文件名，当该文件存在时，则解压指令不执行

- dest：远程主机上的一个路径，即文件解压的路径
- group：解压后的目录或文件的属组
- list_files：如果为yes，则会列出压缩包里的文件，默认为no，2.0版本新增的选项
- mode：解决后文件的权限
- 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 remote_src=yes
```

# 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

    #将/opt/sshkeys目录里所有的文件合并到/root/.ssh/authorized_keys一个文件中
    - name: Build the authorized_keys file
      assemble: src=/opt/sshkeys/ dest=/root/.ssh/authorized_keys owner=root group=root
mode=0700
```