链滴

Ansible 从入门到实战（4）-Ansible 常用模块 A 篇

作者：SmiteLi

原文链接：https://ld246.com/article/1570536783708

来源网站：链滴

Ansible从入门到实战（4）-Ansible常用模块A篇

前面已经初步接触了Ansible的模块，下面对一些常用的模块进行介绍

一、省点力气

下面的几点知识有助于我们学习和理解模块的使用

- 列出ansible所支持的模块。

ansible-doc -l

- 查看模块的详细帮助信息，比如查看 fetch 模块的帮助。

ansible-doc -s fetch

- 调用模块，比如调用 ping模块。

ansible 192.168.128.83 -m ping

- 调用模块的同时传入模块所需要的参数，以 fetch 模块为例。

ansible 192.168.128.83 -m fetch -a "src=/testdir/testfile1 dest=/testdir/ansible/"

二、默认模块commamd

- 作用：在目标主机上执行命令
- 使用例子：

```
chdir # 切换目录
creates # 如果存在，则不执行后面的命令
removes # 如果存在，则执行后面的命令
ansible web -a 'pwd'
ansible web -a 'ls /tmp'
ansible web -a 'creates=/tmp mdkir /data' #被忽略，因为/tmp存在
ansible web -a 'creates=/tmp2 mkdir /data' #被执行，因为/tmp2目录不存在
ansible web -a 'removes=/tmp2 mkdir /data2' #被忽略，因为/tmp2目录不存在
ansible web -a 'removes=/tmp mkdir /data2' # 被执行，因为/tmp存在
```

- 使用实例：

```
ansible apple -a 'creates=/tmp mdkir /data'
```

结果如下：

```
[root@localhost ~]# ansible apple -a 'creates=/tmp mdkir /data'
apple | SUCCESS | rc=0 >>
skipped, since /tmp exists
```

- 注意：

The command(s) will not be processed through the shell, so variables like $HOME and operat ons like "<", ">", "|", ";" and "&" will not work.

三、shell模块

- 作用：在目标主机上执行shell命令
- 使用例子：

```
ansible apple -m shell -a 'echo "1234"|passwd --stdin alex'
ansible apple -m shell -a "bash a.sh" # 执行脚本
ansible apple -m shell -a "/root/a.sh"
ansible apple -m shell -a "/root/a.py" # 执行python文件

ansible apple -m shell -a "ps -ef | grep -E 'api|srv'"


ansible apple -m shell -a "chdir=/data/solution rsync -av  --delete microsso1/ microsso2/"


ansible apple -m shell -a "ps -ef | grep api"
ansible apple -m shell -a "ps -ef | grep srv"
ansible apple -m shell -a "ps -ef | grep -E 'srv|api'"
ansible apple -m shell -a "netstat -lntp | grep -E 'srv|api|apple|consul'"
ansible apple -m shell -a "tail -n10 /data/solution/apple/log/console_output.log"

ansible apple -m shell -a "/data/solution/apple/bin/stop.sh"
ansible apple -m shell -a "tail -n10 /data/solution/apple/log/console_output.log"
ansible apple -m shell -a "df -hT"

ansible apple -m shell -a "ps -ef | grep -E 'srv|api'"
ansible apple -m shell -a "du -h --max-depth=1 /data/"
ansible apple -m shell -a "ls -al /data/solution/apple/page/"
```

- 使用实例：

```
ansible apple -m shell -a 'echo $TERM'
```

结果如下：

```
[root@localhost ~]# ansible apple -m shell -a 'echo $TERM'
apple | CHANGED | rc=0 >>
xterm
```

四、script模块

- 作用：把本地的脚本传到远程目标主机并且执行
- 使用例子：

```
creates 远程主机存在该文件，则跳过
removes 远程主机不存在该文件，则跳过
ansible apple -m script -a "/root/b.sh"
```

```
ansible apple -m script -a "creates=/root/a.py /root/b.sh"
ansible apple -m script -a "removes=/root/a.py /root/b.sh"
```

- 使用实例：

创建/root/b.sh脚本，内容为 ls -l /tmp

执行以下命令

```
ansible apple -m script -a "/root/b.sh"
```

结果如下：

```
[root@localhost ~]# ansible apple -m script -a "/root/b.sh"
apple | CHANGED => {
    "changed": true,
    "rc": 0,
    "stderr": "Shared connection to 192.168.153.129 closed.\r\n",
    "stderr_lines": [
        "Shared connection to 192.168.153.129 closed."
    ],
    "stdout": "total 0\r\ndrwx------ 3 root root 17 Oct  8 15:05 systemd-private-02641de9e2ea
3eb93b950a2213414d1-chronyd.service-94nahM\r\ndrwxr-xr-x 2 root root  6 Oct  8 18:43 tes
\r\ndrwx------ 2 root root  6 Oct  8 15:05 vmware-root_6074-692815760\r\ndrwx------ 2 root
oot  6 Oct  5 01:03 vmware-root_6110-968930893\r\n",
    "stdout_lines": [
        "total 0",
        "drwx------ 3 root root 17 Oct  8 15:05 systemd-private-02641de9e2ea43eb93b950a221
414d1-chronyd.service-94nahM",
        "drwxr-xr-x 2 root root  6 Oct  8 18:43 test",
        "drwx------ 2 root root  6 Oct  8 15:05 vmware-root_6074-692815760",
        "drwx------ 2 root root  6 Oct  5 01:03 vmware-root_6110-968930893"
    ]
}
```

五、copy模块

- 作用：将 ansible 管理主机上的文件拷贝到远程主机中
- 使用例子：

```
backup # 备份
content # 内容
dest # 目的地址
group # 文件的属组
mode #文件的权限  R 4 W 2 X 1
owner # 文件的属主
src #文件的源文件地址
ansible web -m copy -a "dest=/tmp/f src=/etc/fstab" #复制单独文件
ansible web -m copy -a "src=/etc/init.d dest=/tmp"  #复制文件目录
ansible web -m copy -a "src=/etc/init.d/ dest=/tmp" # 复制文件夹内的所有的文件
ansible db -m copy -a "dest=/tmp/b.sh src=/root/b.sh mode=644 owner=alex" # 复制文件并
改权限和属主
ansible db -m copy -a "src=/etc/init.d dest=/tmp owner=alex" # 修改文件夹的权限或者属主属
```

```
时，文件夹内的所有文件的权限也会被修改
# 通过md5来做校验
ansible db -m copy -a "dest=/tmp/b.sh src=/root/b.sh mode=644 owner=alex backup=yes" #
复制文件并将原来的文件做备份
ansible db -m copy -a "dest=/tmp/c.txt content='大弦嘈嘈如急雨，小弦切切如私语'" # 直接往
件里面写入文件，是直接覆盖写入，慎用
```

- 使用实例：

执行以下命令，把目录copy到远程主机

```
ansible apple -m copy -a "src=/etc/init.d dest=/tmp"
```

结果如下：

```
[root@localhost ~]# ansible apple -m copy -a "src=/etc/init.d dest=/tmp"
apple | CHANGED => {
    "changed": true,
    "dest": "/tmp/",
    "src": "/etc/init.d"
}
```

六、fetch模块

- 作用：从远程主机中拉取文件到 ansible 管理主机
- 使用例子：

```
dest #目标地址
src # 源地址
ansible db -m fetch -a "dest=/tmp src=/var/log/cron" # 复制远程被管控机器的文件道管控机器
，以被管控机的ip为目录，并保留原来的目录结构
```

- 使用实例：

执行以下命令，把文件fetch到本地主机

```
ansible apple -m fetch -a "dest=/tmp src=/var/log/cron"
```

结果如下：

```
[root@localhost ~]# ansible apple -m fetch -a "dest=/tmp src=/var/log/cron"
apple | CHANGED => {
    "changed": true,
    "checksum": "f64864c44ef11f161332f5bb2aa7cde186ecebb8",
    "dest": "/tmp/apple/var/log/cron",
    "md5sum": "f6ff5ee1d2700ef7b9f517c83fa48d22",
    "remote_checksum": "f64864c44ef11f161332f5bb2aa7cde186ecebb8",
    "remote_md5sum": null
}
```

七、file模块

- 作用：管理文件和文件属性
- 使用例子：

```
access_time # 创建时间
group # 属组
mode # 权限
owner # 属主
path # 文件的路径
src # 源地址，只有在软连接和硬链接的时候才会使用
state # directory 目录 touch 文件  link 软连接 hard 硬链接 absent 删除
ansible db -m file -a "path=/alex state=directory" # 创建一个目录
ansible db -m file -a "path=/root/alex.txt state=touch" # 创建一个文件
ansible db -m file -a "src=/root/q.txt path=/tmp/q state=link" # 创建软连接，源地址是本机上文件地址
ansible db -m file -a "path=/tmp/q state=absent" # 删除文件或者文件夹
```

- 使用实例：

执行以下命令，把文件fetch到本地主机

``

结果如下：

八、yum模块

- 作用：管理包工具
- 使用例子：

```
disable_gpg_check # 禁止检查gpgcheck
disablerepo # 禁用repo源
enablerepo # 启用repo源
name #包的名称
state  remove 卸载 install 安装
ansible web -m yum -a 'name=python2-pip' # 安装python2-pip
ansible web -m yum -a 'name=python2-pip,redis' #用来安装多个包
ansible web -m yum -a 'name="@Development Tools"' # 用来安装包组
```

- 使用实例：

执行以下命令

```
ansible apple -m yum -a 'name=httpd state=absent'
```

结果如下：

```
[root@localhost ~]# ansible apple -m yum -a 'name=httpd state=absent'
apple | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
```

```
    "changed": false,
    "msg": "",
    "rc": 0,
    "results": [
        "httpd is not installed"
    ]
}
```

从返回结果可以看出我的管理节点上并没有安装httpd

九、service模块

- 作用：管理服务
- 使用例子：

```
name # 包名
enabled # 开机自启动
state  started|stopped|reloaded|restarted
ansible web -m service -a 'name=redis state=started enabled=yes' #启动redis，并设置开机自动
ansible web -m service -a 'name=redis state=stopped' #关闭redis
```

- 使用实例：

执行以下命令

```
ansible apple -m service -a 'name=redis state=started enabled=yes'
```

结果如下：

```
[root@localhost ~]# ansible apple -m service -a 'name=redis state=started enabled=yes'
apple | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "msg": "Could not find the requested service redis: host"
}
```

十、user模块

- 作用：管理用户和用户组
- 使用例子：

```
group #组名
groups # 附加组
home # 家目录位置
remove #删除用户的家目录
shell # 用户登录的shell
system # 系统用户
uid # 用户id
 ansible db -m user -a "uid=500 system=yes groups=root name=mysql" # 创建mysql用户，
```

```
定用户为系统用户，并指定uid为500，指定附加组为root
 ansible db -m user -a "uid=3000 groups=mysql name=canglaoshi home=/opt/canglaoshi sh
ll=/sbin/nologin" # 创建普通用户，并制定用户的家目录和登录shell以及uid，附加组
 ansible db -m user -a "name=canglaoshi remove=yes state=absent" # 删除用户
```

- 使用实例：

执行以下命令

```
ansible apple -m user -a "name=canglaoshi remove=yes state=absent"
```

结果如下：

```
[root@localhost ~]# ansible apple -m user -a "name=canglaoshi remove=yes state=absent"
apple | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "name": "canglaoshi",
    "state": "absent"
}
```

十一、下一节是?

下一节继续介绍常用的Ansible模块