29 Ansible模块

Ansible模块

一、模块概述

Ansible Ad-Hoc 模块一般称之为**模块**或**命令**。

Ad-Hoc英文中作为形容词有“特别的，**临时**”的含义。

从功能上讲，Ad-Hoc是相对Playbook而言的，Ansible提供两种完成任务方式：Ad-Hoc模块和Playbook剧本。

- **模块**更注重于解决一些简单或者平时工作中**临时**遇到的任务，相当于**一个**Linux系统命令行下的**Shell命令**。
- **Playbook**更适合于解决复杂或需固化下来的任务，**相当于Linux系统的Shell脚本**。

模块可通过 `ansible` 命令直接运行。

`ansible` 命令常用的语法格式为 `ansible 主机/组 [选项]` 。常见选项如下表所示。

选项	作用
-i	指定主机清单文件
-m	指定要使用的模块名
-a	设置传递给模块的参数（属性）
--list-hosts	列出符合条件的主机列表，不执行任何命令
-u	**用户**
-k	**手动输入SSH协议密码**
-M	指定要使用的模块路径
-T	设置SSH协议连接超时时间
--version	查看版本信息
-h	帮助信息
-o	**简洁输出**

ansible运行模块的常用语法格式为 **`ansible 主机/组 -m 模块名称 [-a模块参数]`** 。

`ansible 主机/组 -m 模块名称 [-a 模块参数] -u 用户名 -k`

下面以一些简单的案例说明ansible如何运行模块。

准备工作

Shell | 复制代码

```
[root@zabbix ~]# yum install ansible -y --nogpgcheck
# 取消指纹验证
[root@zabbix ~]# sed -i 's/#host_key_checking/host_key_checking/' /etc/ansible/ansible.cfg
# 设置inventory
[root@zabbix ~]# vi /etc/ansible/hosts
192.168.149.4
[test]
192.168.149.3
[test:vars]
ansible_user="root"
ansible_password="000000"
```

案例：使用 `ping` 模块检查test组所有主机是否存活。

Shell | 复制代码

```
[root@zabbix ~]# ansible test -m ping
192.168.149.3 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

- 192.168.149.3是指命令执行的主机。
- Success表示命令执行成功。
- “=>{}”表示详细返回结果。
- "changed": false表示没有对主机做变更。
- "ping": "pong"表示执行了ping命令返回结果为pong。

案例：使用 `command` 模块返回test组所有主机的hostname。

```shell
[root@zabbix ~]# ansible test  -m command -a "hostname"
192.168.149.3 | CHANGED | rc=0 >>
shell
```

使用 `-vvv` 参数可以清楚地了解Ansible命令执行流程。

```shell
[root@zabbix ~]# ansible test  -m command -a "hostname" -vvv
```

通过前面的案例可以发现，模块的功能较为强大，Ansible自身提供了大量模块，可以通过 `ansible-doc` 命令查看相关模块的信息。

```shell
# 列出内置模块的列表
[root@zabbix ~]# ansible-doc -l
# 统计内置模块的数量
[root@zabbix ~]# ansible-doc -l | wc -l
3387
# 查看模块帮助的语法为ansible-doc 命令名
# 查看command模块的帮助
[root@zabbix ~]# ansible-doc command
```

ansible模块列表及帮助信息详见
https://docs.ansible.com/ansible/2.9/modules/list_of_all_modules.html#all-modules

ansible常用的内置模块如下。

模块名称	模块作用
ping	检查受管节点主机网络是否能够连通。SSH是否正常。
yum	安装、更新及卸载软件包。
yum_repository	管理主机的软件仓库配置文件。
template	复制模板文件到受管节点主机。
copy	新建、修改及复制文件。
user	创建、修改及删除用户。
group	创建、修改及删除用户组。

service	**启动、关闭及查看服务状态。**
get_url	从网络中下载文件。
file	**设置文件权限及创建快捷方式。**
cron	添加、修改及删除计划任务。
command	**直接执行用户指定的命令。**
shell	**直接执行用户指定的命令（支持特殊字符）。**
debug	输出调试或报错信息。
mount	挂载硬盘设备文件。
filesystem	格式化硬盘设备文件。
lineinfile	**通过正则表达式操作文件内容。**
setup	收集受管节点主机上的系统及变量信息。
firewalld	添加、修改及删除防火墙策略。
lvg	管理主机的物理卷及卷组设备。
lvol	管理主机的逻辑卷设备。

二、模块应用实例

2.1 setup模块

facts组件是Ansible用来采集**客户端机器设备信息**的一个重要功能，`setup`模块可用于获取Ansible客户端机器的所有facts信息，并且可以通过filter属性来查看指定的信息。

案例：显示test组主机的facts信息

Shell | 复制代码

```
[root@zabbix ~]# ansible test -m setup | more
192.168.149.3 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "192.168.149.3"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::20c:29ff:fe5a:ab94"
        ],
        "ansible_apparmor": {
            "status": "disabled"
        },
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "07/02/2015",
        "ansible_bios_version": "6.00",
        "ansible_cmdline": {
            "BOOT_IMAGE": "/vmlinuz-3.10.0-327.el7.x86_64",
            "LANG": "en_US.UTF-8",
            "crashkernel": "auto",
            "quiet": true,
            "rd.lvm.lv": "centos/swap",
            "rhgb": true,
            "ro": true,
            "root": "/dev/mapper/centos-root"
        },
        "ansible_date_time": {
            "date": "2022-05-24",
            "day": "24",
            "epoch": "1653385922",
            "hour": "17",
            "iso8601": "2022-05-24T09:52:02Z",
            "iso8601_basic": "20220524T175202965149",
            "iso8601_basic_short": "20220524T175202",
            "iso8601_micro": "2022-05-24T09:52:02.965149Z",
            "minute": "52",
            "month": "05",
            "second": "02",
```

案例：通过filter参数指定显示IPv4地址

```
[root@zabbix ~]# ansible test -m setup  -a "filter=ansible_all_ipv4_addresses"
192.168.149.3 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "192.168.149.3"
        ],
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false
}
```

2.2 command模块

`command` 模块是Ansible的**默认模块**，在使用时可以**省略** `-m command`， `command` 模块的功能是执行指定的shell命令。

案例：批量查看test组所有主机的磁盘容量

```
#两种命令格式效果相同
[root@zabbix ~]# ansible test -a "df -h"
192.168.149.3 | CHANGED | rc=0 >>
Filesystem               Size  Used Avail Use% Mounted on
/dev/mapper/centos-root   18G  2.2G   16G  13% /
devtmpfs                 479M     0  479M   0% /dev
tmpfs                    489M     0  489M   0% /dev/shm
tmpfs                    489M  6.7M  483M   2% /run
tmpfs                    489M     0  489M   0% /sys/fs/cgroup
/dev/sda1                497M  125M  373M  25% /boot
tmpfs                     98M     0   98M   0% /run/user/0
[root@zabbix ~]# ansible test -m command -a "df -h"
192.168.149.3 | CHANGED | rc=0 >>
Filesystem               Size  Used Avail Use% Mounted on
/dev/mapper/centos-root   18G  2.2G   16G  13% /
devtmpfs                 479M     0  479M   0% /dev
tmpfs                    489M     0  489M   0% /dev/shm
tmpfs                    489M  6.7M  483M   2% /run
tmpfs                    489M     0  489M   0% /sys/fs/cgroup
/dev/sda1                497M  125M  373M  25% /boot
tmpfs                     98M     0   98M   0% /run/user/0
```

2.3 shell模块

`shell` 模块与 `command` 模块类似，但是 **`shell` 模块支持管道符，还用于执行被控端的Shell脚本文件**。

案例： `shell` 模块和 `command` 模块对比

Bash 复制代码

```
[root@zabbix ~]# ansible test -m shell -a "cat /etc/passwd|wc -l"
192.168.149.3 | CHANGED | rc=0 >>
23
[root@zabbix ~]# ansible test -m command -a "cat /etc/passwd|wc -l"
192.168.149.3 | FAILED | rc=1 >>
cat: invalid option -- 'l'
Try 'cat --help' for more information.non-zero return code
```

案例：批量查看远程主机内存使用情况

Shell 复制代码

```
[root@zabbix ~]# ansible test -m shell -a "free -h"
192.168.149.3 | CHANGED | rc=0 >>
              total        used        free      shared  buff/cache   available
Mem:           977M        211M        587M        7.0M        178M        604M
Swap:          2.0G          0B        2.0G
```

案例：执行test组机器下的 `/root/test.ab` 文件

Shell 复制代码

```
[root@shell ~]# vi /root/test.ab
echo abc
[root@shell ~]# . /root/test.ab
abc
[root@zabbix ~]# ansible test -m shell -a ". /root/test.ab"
192.168.149.3 | CHANGED | rc=0 >>
abc
```

2.4 script模块

`script` 模块用于在远程**被控端主机执行主控端**中的Shell脚本文件，相当于 `scp+shell` 的组合命令。

```shell
# 控制节点新建脚本文件
[root@zabbix ~]# vi s.sh
touch 123
# 在test组主机上运行s.sh
# 注意，原来test组主机上是没有s.sh的
[root@zabbix ~]# ansible test -m script -a "/root/s.sh"
192.168.149.3 | CHANGED => {
    "changed": true,
    "rc": 0,
    "stderr": "Shared connection to 192.168.149.5 closed.\r\n",
    "stderr_lines": [
        "Shared connection to 192.168.149.5 closed."
    ],
    "stdout": "",
    "stdout_lines": []
}
[root@zabbix ~]# ansible test -a "ls /root/"
192.168.149.3 | CHANGED | rc=0 >>
123
```

Ansible会将主控端下的/root/s.sh分别PUT到被控端机器的/root/.ansible/tmp/临时目录下，然后再分别执行并删除。

2.5 copy模块

`copy` 模块可实现Ansible主机向客户端传送文件的功能，文件的变化是通过**md5**值来判断的，**注意应提前关闭受控端的SELinux，不然会出现错误**。

- `force` 参数：如果目标主机包含该文件，但内容不同，则设置为 `yes` 后会强制覆盖，设置为 `no` 时，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` 。
- `backup` 参数：在覆盖之前备份源文件，备份文件包含时间。 `backup` 包含两个选项，即 `yes` 和 `no` 。

如果路径使用 `/` 来结尾，则只需要复制目录里的内容；如果没有使用 `/` 来结尾，则包含目录和文件在内的整个内容全部都要复制（源目标目录作为目的目录的一个子目录存在）。

案例：将主控端的/root/ip.txt复制为受控端的/root/ip2.txt，并设置权限

```shell
# 将主控端的/root/ip.txt复制为受控端的/root/ip2.txt，并设置权限
[root@zabbix ~]# ansible test -m copy -a "src=/root/ip.txt dest=/root/ip2.txt owner=root group=root mode=0755 force=yes"
192.168.149.3 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "checksum": "5fda085407a56e965f9ba7582841ea9607329072",
    "dest": "/root/ip2.txt",
    "gid": 0,
    "group": "root",
    "md5sum": "5a0d7b3aefa3eb20ac1629c14a17509a",
    "mode": "0755",
    "owner": "root",
    "secontext": "system_u:object_r:admin_home_t:s0",
    "size": 13,
    "src": "/root/.ansible/tmp/ansible-tmp-1653378968.02-34090-228627109580021/source",
    "state": "file",
    "uid": 0
}
# 检查运行结果
[root@zabbix ~]# ansible test -a "ls /root/ip2.txt"
192.168.149.3 | CHANGED | rc=0 >>
/root/ip2.txt
```

2.6 file模块

`file` 模块主要用来设置文件或目录的属性。常用参数如下。

- group：定义文件或目录的属组。
- mode：定义文件或目录的权限。
- owner：定义文件或目录的属主。
- **path：必选项，定义文件或目录的路径。**
- recurse：递归设置文件的属性，只对目录有效。
- **src：被链接的源文件路径，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。**
- **force：强制创建软链接包含两种情况，一种是源文件不存在，但之后会建立的情况；另一种是要取消已创建的软链接，创建新的软链接，其包含yes和no这两个选项。**

- state：后面连接文件或目录的各种状态。
- link：创建软链接。
- hard：创建硬链接。
- directory：如果目录不存在，则创建目录。
- file：即使文件不存在，也不会被创建。
- absent：删除目录、文件或链接文件。
- 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，则更新其最后的修改时间，这一点与Linux的touch命令的效果是一样的。

案例：将主控端的 `/root/ip.txt` 软链接到 `/tmp/ip.txt` 下

Shell | 复制代码

```
[root@zabbix ~]# ansible test -m file -a "src=/root/ip.txt dest=/tmp/ip.txt state=link"
192.168.149.3 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "dest": "/tmp/ip.txt",
    "gid": 0,
    "group": "root",
    "mode": "0777",
    "owner": "root",
    "secontext": "unconfined_u:object_r:user_tmp_t:s0",
    "size": 12,
    "src": "/root/ip.txt",
    "state": "link",
    "uid": 0
}
[root@zabbix ~]# ansible test -a "ls /tmp/ip.txt"
192.168.149.3 | CHANGED | rc=0 >>
/tmp/ip.txt
```

案例： 删除刚刚建立的 `/tmp/ip.txt` 链接文件

```
[root@zabbix ~]# ansible test -m file -a "path=/tmp/ip.txt state=absent"
192.168.149.3 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "path": "/tmp/ip.txt",
    "state": "absent"
}
# 这里显示报错信息，表示此文件不存在，说明已经成功将其删除了。
[root@zabbix ~]# ansible test -a "ls /tmp/ip.txt"
192.168.149.5 | FAILED | rc=2 >>
ls: cannot access /tmp/ip.txt: No such file or directorynon-zero return co
de
```

案例：在test组建立 `/root/text.txt` 文件，属主和属组均为root，权限为0755

```
[root@zabbix ~]# ansible test -m file -a "path=/root/text.txt owner=root g
roup=root mode=0755 state=touch"
192.168.149.3 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "dest": "/root/text.txt",
    "gid": 0,
    "group": "root",
    "mode": "0755",
    "owner": "root",
    "secontext": "unconfined_u:object_r:admin_home_t:s0",
    "size": 0,
    "state": "file",
    "uid": 0
}
```

案例：在test组建立 `/tmp/test` 目录，属主和属组均为root，权限为0755

```shell
[root@zabbix ~]# ansible test -m file -a "path=/tmp/test owner=root group=root mode=0755 state=directory"
192.168.149.3 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "gid": 0,
    "group": "root",
    "mode": "0755",
    "owner": "root",
    "path": "/tmp/test",
    "secontext": "unconfined_u:object_r:user_tmp_t:s0",
    "size": 6,
    "state": "directory",
    "uid": 0
}
```

2.7 user模块

Ansible系统用户模块有如下两个：Linux系统用户管理 `user` 和Windows系统用户管理 `win_user` 。

案例：新增用户dba，使用BASH Shell，附加组为root

```
[root@zabbix ~]# ansible test -m user -a "name=dba shell=/bin/bash groups=root append=yes  state=present"
192.168.149.3 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "comment": "",
    "create_home": true,
    "group": 1001,
    "groups": "root",
    "home": "/home/dba",
    "name": "dba",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 1001
}
```

案例：删除用户dba

```
[root@zabbix ~]# ansible test -m user -a "name=dba remove=yes  state=absent"
192.168.149.3 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "append": false,
    "changed": false,
    "comment": "",
    "group": 1001,
    "home": "/home/dba",
    "move_home": false,
    "name": "dba",
    "shell": "/bin/bash",
    "state": "absent",
    "uid": 1001
}
```

2.8 group模块

`group` 模块可以在所有节点上创建自己定义的组。

案例：创建组名为test、gid为2018的组

```shell
[root@zabbix ~]# ansible test -m group -a "gid=2018 name=test"
192.168.149.3 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "gid": 2018,
    "name": "test",
    "state": "present",
    "system": false
}
#验证
[root@zabbix ~]# ansible test -m shell -a "cat /etc/group | grep test"
192.168.149.3 | CHANGED | rc=0 >>
test:x:2018:
```

2.9. service

`service` 模块用于被控端服务管理，例如开启、关闭、重启服务等。

- state参数：控制服务状态，属性值为reloaded/restarted/started/stopped。
- enabled参数：控制服务是否自启动，属性值为：yes/no。

案例：操作zabbix-agent服务

Shell | 复制代码

```
# 控制test组主机启动zabbix-agent服务
[root@zabbix ~]# ansible test -m service -a "name=zabbix-agent state=started"
192.168.149.3 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "name": "zabbix-agent",
    "state": "started",
# 控制test组主机重启zabbix-agent服务
[root@zabbix ~]# ansible test -m service -a "name=zabbix-agent state=restarted"
192.168.149.3 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "name": "zabbix-agent",
    "state": "started",
# 控制test组主机停止zabbix-agent服务
[root@zabbix ~]# ansible test -m service -a "name=zabbix-agent state=stopped"
192.168.149.3 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "name": "zabbix-agent",
    "state": "stopped",
# 查看test组主机zabbix-agent服务状态
[root@zabbix ~]# ansible test -a "systemctl status zabbix-agent"
192.168.149.3 | FAILED | rc=3 >>
● zabbix-agent.service - Zabbix Agent
   Loaded: loaded (/usr/lib/systemd/system/zabbix-agent.service; disabled; vendor preset: disabled)
   Active: inactive (dead)

May 24 18:26:31 database systemd[1]: Starting Zabbix Agent...
May 24 18:26:31 database systemd[1]: PID file /run/zabbix/zabbix_agentd.pid not readable (yet?) after start.
May 24 18:26:31 database systemd[1]: Started Zabbix Agent.
May 24 18:26:55 database systemd[1]: Stopping Zabbix Agent...
May 24 18:26:55 database systemd[1]: Starting Zabbix Agent...

```

```
May 24 18:26:56 database systemd[1]: PID file /run/zabbix/zabbix_agentd.pi
d not readable (yet?) after start.
May 24 18:26:56 database systemd[1]: Started Zabbix Agent.
May 24 18:27:49 database systemd[1]: Stopping Zabbix Agent...
May 24 18:27:49 database systemd[1]: Stopped Zabbix Agent.non-zero return
code
# 控制test组主机启动zabbix-agent服务，并设置开机自启动
[root@zabbix ~]# ansible test -m service -a "name=zabbix-agent state=start
ed enabled=yes"
192.168.149.3 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "enabled": true,
    "name": "zabbix-agent",
    "state": "started",
    "status": {
```

2.10 yum模块

yum模块可使用yum包管理器管理软件包。

- name参数：需要操作的软件包的名称，可带版本或包说明符。
- state参数：管理软件包，属性值为absent/installed/latest/present/removed。present和installed将安装所需的软件包，latest将更新指定的包，absent和removed将删除指定的包。
- disable_gpg_check 参数：是否对包进行GPG签名检查，属性值为yes/no，默认为no。

案例：通过yum模块安装locate命令

Bash | 复制代码

```
# 验证
[root@shell ~]# locate
-bash: locate: command not found
# 安装locate
[root@zabbix ~]# ansible test -m yum -a "name=mlocate state=installed"
192.168.149.3 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "changes": {
        "installed": [
            "mlocate"
        ]
    },
    "msg": "",
    "rc": 0,
    "results": [
        "Loaded plugins: fastestmirror\nLoading mirror speeds from cached hostfile\nResolving Dependencies\n--> Running transaction check\n---> Package mlocate.x86_64 0:0.26-8.el7 will be installed\n--> Finished Dependency Resolution\n\nDependencies Resolved\n\n================================================================================\n Package          Arch          Version               Repository         Size\n================================================================================\nInstalling:\n mlocate          x86_64          0.26-8.el7            opt_soft          113 k\n\nTransaction Summary\n================================================================================\nInstall  1 Package\n\nTotal download size: 113 k\nInstalled size: 379 k\nDownloading packages:\nRunning transaction check\nRunning transaction test\nTransaction test succeeded\nRunning transaction\n  Installing : mlocate-0.26-8.el7.x86_64                                   1/1 \n  Verifying  : mlocate-0.26-8.el7.x86_64                                   1/1 \n\nInstalled:\n  mlocate.x86_64 0:0.26-8.el7                                              \n\nComplete!\n"
    ]
}
# 验证
[root@shell ~]# locate
locate: no pattern to search for specified
# 卸载locate
[root@zabbix ~]# ansible test -m yum -a "name=mlocate state=removed"
192.168.149.3 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
```

```
    "changed": true,
    "changes": {
        "removed": [
            "mlocate"
        ]
    },
    "msg": "",
    "rc": 0,
    "results": [
        "Loaded plugins: fastestmirror\nResolving Dependencies\n--> Running transaction check\n---> Package mlocate.x86_64 0:0.26-8.el7 will be erased\n--> Finished Dependency Resolution\n\nDependencies Resolved\n\n================================================================================\n Package          Arch            Version              Repository          Size\n================================================================================\nRemoving:\n mlocate          x86_64          0.26-8.el7           @opt_soft          379 k\n\nTransaction Summary\n================================================================================\nRemove  1 Package\n\nInstalled size: 379 k\nDownloading packages:\nRunning transaction check\nRunning transaction test\nTransaction test succeeded\nRunning transaction\n  Erasing    : mlocate-0.26-8.el7.x86_64                                  1/1 \n  Verifying  : mlocate-0.26-8.el7.x86_64                                  1/1 \n\nRemoved:\n  mlocate.x86_64 0:0.26-8.el7                                                \n\nComplete!\n"
    ]
}
# 验证
[root@shell ~]# locate
-bash: /usr/bin/locate: No such file or directory
```

2.11 lineinfile模块

lineinfile模块的功能是对文件中的行内容进行操作，**类似于sed命令**。

- path：指定要操作的文件。
- regexp：使用正则表达式匹配对应的行
- line：匹配行的新内容
- insertafter：将文本插入到匹配行之后
- insertbefore：将文本插入到匹配行之前
- state：删除对应的文本时，需要state=absent
- create：当要操作的文件并不存在时，是否创建对应的文件
- backup： 是否备份原文件，属性值为yes/no，默认为no

- backrefs： 属性值为yes/no，默认为no
  - 当backrefs为no时，如果regex没有匹配到行，则添加一行，如果regex匹配到行，则修改该行
  - 当backrefs为yes时，如果regex没有匹配到行，则保持原文件不变，如果regex匹配到行，则修改该行

案例：修改文件内容

Bash | 复制代码

```
[root@zabbix ~]# ansible test -m command -a  "sed -n '/^SELINUX=/p' /etc/selinux/config"
[WARNING]: Consider using the replace, lineinfile or template module rather than running 'sed'.  If you need to use command
because replace, lineinfile or template is insufficient you can add 'warn: false' to this command task or set
'command_warnings=False' in ansible.cfg to get rid of this message.
192.168.149.3 | CHANGED | rc=0 >>
SELINUX=enforcing
[root@zabbix ~]# ansible test -m lineinfile -a 'path=/etc/selinux/config regexp="^SELINUX=" line="SELINUX=disabled"'
192.168.149.3 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "backup": "",
    "changed": true,
    "msg": ""
}
[root@zabbix ~]# ansible test -m command -a  "sed -n '/^SELINUX=/p' /etc/selinux/config"
[WARNING]: Consider using the replace, lineinfile or template module rather than running 'sed'.  If you need to use command
because replace, lineinfile or template is insufficient you can add 'warn: false' to this command task or set
'command_warnings=False' in ansible.cfg to get rid of this message.
192.168.149.3 | CHANGED | rc=0 >>
SELINUX=disabled
```

2.12 get_url模块

`get_url` 模块可以实现在远程主机上下载url到本地。

- url参数：待获取的url地址。
- dest参数：下载文件的存放位置。

案例：下载bc软件包

```shell
# test组主机远程获取bc安装包，并保存在/tmp目录
[root@zabbix ~]# ansible test -m get_url -a "url=http://mirror.centos.org/centos/7/os/x86_64/Packages/bc-1.06.95-13.el7.x86_64.rpm dest=/tmp"
# 验证
[root@zabbix ~]# ansible test -a "ls /tmp"
192.168.149.3 | CHANGED | rc=0 >>
ansible_command_payload_rG4eKN
bc-1.06.95-13.el7.x86_64.rpm
```

小结

`ansible 主机/组 -m 模块 -a "模块参数"`

课程目标

- 知识目标：了解模块的特性、模块的基本用法。
- 技能目标：能够根据需求使用模块完成临时性任务。

课外拓展

- 了解更多Ansible模块的应用场景。

参考资料

- Ansible官方文档：https://docs.ansible.com/
- https://docs.ansible.com/ansible/2.9/modules/list_of_all_modules.html#all-modules
- 《DevOps和自动化运维实践》，余洪春，机械工业出版社
- 《Ansible权威指南》，李松涛，机械工业出版