

Hello Ansible Live Guideline

1 Environment Settings


1.1 Create VM for everyone

- Get the numbers of participants who need to practise ansible. create the ECS in [aliyun cloud](#)
- Get the Public IP and Private IP. Assign them to Participants. Export the result to excel.
- Group the private ip. for example:

实例ID	名称	公网IP	内网IP	状态
			[group1]	
i-uf6i6a4xufjjch4ouyd6	master	139.224.198.140	10.0.1.54	Running
i-uf6i6a4xufjjch4ouyd7	master	101.132.108.10	10.0.1.53	Running
			[group2]	
i-uf652prtht947k8x7ms6	master	47.101.39.205	10.0.1.51	Running
i-uf652prtht947k8x7ms5	master	139.196.205.228	10.0.1.52	Running

1.2 Clone github repository

- Introduce the github repository. use the Zip as backup. In case participants can not clone it.
- `git clone git@github.com:Sirius0301/hello_ansible_public.git`

 MINGW64:/c/Users/siriu/Desktop

```
siriu@LAPTOP-Q7E7NRV0 MINGW64 ~/Desktop
$ git clone git@github.com:Sirius0301/hello_ansible.git
Cloning into 'hello_ansible'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
Receiving objects: 37% (3/8), 316.00 KiB | 17.00 KiB/s
```

Notice

- Minimum required git version is 2.18. please run `git --version` to check

Minimum required git version is 2.18. Your git ('/usr/bin/git') is 2.17.1 #88

kevinburkemeter opened this issue on Dec 4, 2019 · 1 comment



kevinburkemeter commented on Dec 4, 2019

Just posting in case anyone runs into this and looks in the wrong place. The actions/checkout step failed with this output:

```
/usr/bin/docker exec 2a47320f8dc42ac77bde007d1695d03019f1ca5eb41d19d4bf6816053421c0 sh -c "cat /etc/*release | grep ^ID"
Running JavaScript Action with default external tool: node12
Added matchers: 'checkout-git'. Problem matchers scan action output for known warning or error strings and report these in.
Syncing repository: owner/project
Working directory is '/__w/project/project'
/usr/bin/git version
git version 2.17.1
Removed matchers: 'checkout-git'
##[error]Minimum required git version is 2.18. Your git ('/usr/bin/git') is 2.17.1
##[error]Node run failed with exit code 1
```

The problem was I was using a custom Docker container with an outdated Git version installed. Update the Git version in the custom container and it should work again.



1



kevinburkemeter closed this on Dec 4, 2019

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a p
issue.

None yet

Notifications



- Basic authentication using a password to Git is deprecated and will soon no longer work.
 - using ssh-keygen to add ssh key into your github account
- for example: `ssh-keygen -t rsa -b 4096 -C "sirius.wf.wang@gmail.com"`

[GitHub] Deprecation Notice

sirius.wf.wang@gmail.com



GitHub 2021-03-20 16:31

发至 我



详情

Hi @Sirius0301,

You recently used a password to access the repository at Sirius0301/hello_ansible with git using git/2.31.0.windows.1.

Basic authentication using a password to Git is deprecated and will soon no longer work. Visit <https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/> for more information around suggested workarounds and removal dates.

Thanks,
The GitHub Team

- if there is error like below, please change the privilege code of your ssh private key file.
 - in linux or mac you can run `sudo chmod 600 .ssh/id_rsa`

```
ubuntu@VM-0-2-ubuntu:~$ git clone git@github.com:Sirius0301/hello_ansible.git
Cloning into 'hello_ansible'...
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@                WARNING: UNPROTECTED PRIVATE KEY FILE!                @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0664 for '/home/ubuntu/.ssh/id_rsa' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "/home/ubuntu/.ssh/id_rsa": bad permissions
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
```

you can get more about github ssh information [github docs - about ssh](#)

2 Introduce Ansible by using the PPT as guideline

2.1 环境准备

参与者应提前准备 xshell or putty

2.2 Ansible 应用场景

见 ppt [5,8] 页

- 开发场景 -- Toby
- 运维场景 -- Toby
- Citi 应用场景 -- Toby

2.3 发展历史

见 ppt 9页

- 发展历史

3 Ansible 讲解

3.1 安装 ansible

手动安装ansible到个人使用主机, 使用脚本 initial_ansible.sh

```
cd hello_ansible/  
  
chmod +x shell/initial_ansible.sh  
  
yes Yes| shell/initial_ansible.sh
```

or

```
yum install ansible  
ansible --version
```

yum install git

顺便介绍 Linux 权限code与 修改权限code 命令

- 表示文件 D 表示目录				U: owner			G: member in the group			O: everyone else		
- Or D	r	w	x	r	w	x	r	w	x	r	w	x
	4	2	1	4	2	1	4	2	1	4	2	1

chmod +x initial_ansible.sh <=> chmod 700 initial_ansible.sh

3.2 尝试运行ansible

执行如下ansible 命令和 ansible-playbook 命令，会出现错误

```
ansible group1 -m ping
```

```
ansible-playbook playbook_01.yml
```

```
#playbook01.yml
---
#test connection yaml file
- hosts: group1
  remote_user: root
  tasks:
    - name: test connection
      ping:
    ...
```

```
[root@master home]# ansible group1 -m ping
10.0.1.54 | UNREACHABLE! => {
  "changed": false,
  "msg": "Failed to connect to the host via ssh: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).",
  "unreachable": true
}
10.0.1.53 | UNREACHABLE! => {
  "changed": false,
  "msg": "Failed to connect to the host via ssh: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).",
  "unreachable": true
}
[root@master home]# ansible-playbook playbook_01.yml
PLAY [group1] *****
TASK [Gathering Facts] *****
fatal: [10.0.1.54]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect to the host via ssh: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).", "unreachable": true}
fatal: [10.0.1.53]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect to the host via ssh: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).", "unreachable": true}
PLAY RECAP *****
10.0.1.53 : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ignored=0
10.0.1.54 : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ignored=0
```

因为 ansible是基于ssh, 所以必须要和ssh一样，远程连接，指定用户，并输入密码

```
ansible group1 -m ping -u root -k
```

input your password, then you will see:

```
[root@master home]# ansible group1 -m ping -u root -k
SSH password:
10.0.1.54 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
10.0.1.53 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
```

3.3 ssh 配置免密登录

生成 公钥密钥

```
ssh-keygen -t dsa
```

拷贝公钥到node节点

```
ssh-copy-id -i /root/.ssh/id_dsa.pub root@10.0.1.54
```

验证 node 是否已添加

```
ssh root@10.0.1.54
```

```
ll /root/.ssh/authorized_keys
```

批量复制 id_dsa.pub 到node 节点

shell/copy_ssh_pub_to_all_hosts.sh

```
for server in `cat hosts.txt`;
do
    sshpass -p "!!QAZ2wsx" ssh-copy-id -i ~/.ssh/id_dsa.pub root@$server
done
```

3.4 ansible 配置文件介绍

```
vim /etc/ansible/ansible.cfg
```

```
[defaults]
inventory = /etc/ansible/hosts          #主机列表配置文件
library = /usr/share/my_modules          #库文件存放目录
remote_tmp = $HOME/.ansible/tmp          #临时py命令存放在远程主机目录
local_tmp = $HOME/.ansible/tmp           #本机的临时命令执行目录
forks = 5                                #默认并发数
sudo_user = root                         #默认sudo用户
ask_sudo_pass = True                     #主机列表配置文件
remote_port = 22
host_key_checking = False                #检查对应服务器的host_key
log_path = /var/log/ansible.log          #日志文件，建议启用
module_name = command                    #默认模块，可以修改为shell模块
```

打开log 日志

```
tail -20 /var/log/ansible.log
```

vim 设置行号可见

```
:set nu
```

3.5 ansible 主机清单

3.5.1 主机清单

手动编辑 /etc/ansible/hosts里面的内容

```
vim /etc/ansible/hosts
```

```
ansible all --list-hosts
```

3.5.2 ansible 的主机清单匹配模式

- All : 表示所有Inventory 主机

```
ansible all -m ping
```

- ◦ : 通配符

```
ansible "*" -m ping
```

```
ansible 192.168.1.* -m ping
```

```
ansible "*srvs" -m ping
```

- 逻辑或

```
ansible "websrvs:appsrvs" -m ping
```

```
ansible "192.168.1.10:192.165.1.20" -m ping
```

- 逻辑与

在websrvs组 并且在 dbsrvs 组中的主机

```
ansible "websrvs:&dbsrvs" -m ping
```

- 逻辑非

- 正则表达式

```
ansible "~(web | db)srvs" -m ping
```

```
ansible "~(web | db).*\.magedu\.com" -m ping
```

- ansible-galaxy

下载优秀的roles模板 [ansible-galaxy](https://galaxy.ansible.com/).

```
ansible-galaxy list
```

3.5 ansible 特性

3.5.1 基于ssh, 也因此是 serveless的服务

见ppt 10页

- 利用ssh 远程执行 shell脚本

```
scp -p22 -r -p hello_ansible/shell/initial_ansible.sh root@10.0.1.54:/home/
```

```
ssh root@10.0.1.54
```

```
chmod +x /home/initial_ansible.sh
```

```
/home/initial_ansible.sh
```

- 利用ansible-playbook 执行一样的操作

```
ansible-playbook playbook_02.yml
```

```
#playbook_02.yml
- hosts: 10.0.1.54
  remote_user: root
  tasks:
    - name: copy initial_ansible.sh
      copy:
        src: /home/hello_ansible/shell/initial_ansible.sh
        dest: /home
        mode: 700
    - name: excute initial_ansible.sh
      shell:
        cmd: yes Yes | /home/initial_ansible.sh
        executable: /bin/bash
```

3.5.2 ansible 幂等性

多次执行

```
ansible-playbook playbook_02.yml
```

结果一致

3.5.3 Yaml 语法

- 以 --- 开头, 以 ... 结尾, 可省略
- #表示注释, 且仅支持单行注释
- 表示列表类型, 也代表局部作用域
- 严格的缩进格式和缩进级别, 建议统一采用空格作为缩进
- 字符串通常不添加任何引号, 即使其包含特殊字符

4 ansible-playbook

4.1 查看帮助命令

```
ansible-playbook --help
```

4.2 检查 playbook.yml 语法

```
ansible-playbook -C playbook_03.yml
```

```
ansible-playbook --syntax-check playbook_03.yml
```

4.3 ansible-playbook 基础

ansible module

4.3.1 查看ansible 目前总的 module 数量

```
ansible-doc -l | wc -l
```

4.3.2 常用module 总结

- copy
- scp
- command
- script
- template
- user
- git
- cron
- file
- group
- service
- yum
-

根据实际应用与生产需要，逐个模块使用学习和熟悉

5 playbook 中关键字

5.1 facts

Ansible 使用facts 在被托管主机上自动收集的变量，通过执行 ansible命令以及playbook时 setup模块起效。该操作时默认的。

```
ansible 10.0.1.54 > ansible_facts.json
```

```
ansible-playbook playbook_04.yml
```

```
#playbook_04.yml
- hosts: 10.0.1.54
  tasks:
    - debug:
        var: ansible_facts
```

playbook 中可使用 `gather_facts = no` 禁止，会加快ansible的执行效率

收集信息包括：主机名、内核版本、网卡接口、IP 地址、操作系统版本、环境变量、CPU 核数、可用内存、可用磁盘 等等

详见 `ansible_facts.json`

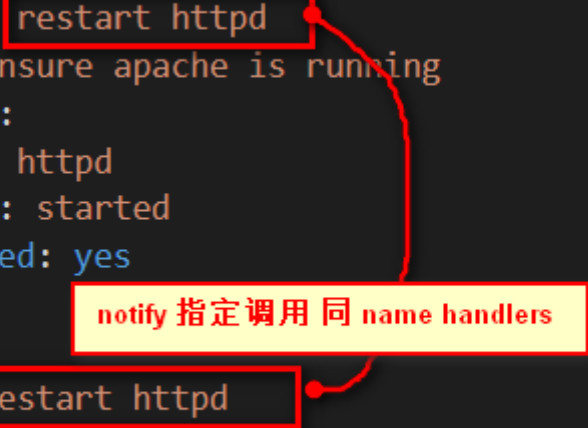
```
ansible-playbook playbook_05.yml
```

```
#playbook_05.yml
- hosts: 10.0.1.54
  gather_facts: no
  tasks:
    - debug:
        var: ansible_facts
```


5.2 notify & handler

与handler 成对出现，调用handler下的task. 在对应的task 最后被出发，可用于playbook 多个task 执行完，需要一次性指定操作时使用

```
1  ---
2  - hosts: group1
3    remote_user: root
4    gather_facts: no
5
6    tasks:
7      - name: Install httpd
8        yum:
9          name: httpd
10         state: present
11      - name: Install configure file
12        copy:
13          src: /home/hello_ansible/shell/httpd.conf
14          dest: /etc/httpd/conf/
15          notify: restart httpd
16      - name: ensure apache is running
17        service:
18          name: httpd
19          state: started
20          enabled: yes
21
22    handlers:
23      - name: restart httpd
24        service: name=httpd state=restarted
25  ...
```



notify 指定调用 同 name handlers

ansible-playbook playbook_03.yml

5.3 tags

标签的主要作用时可以在ansible-playbook中设置只执行哪些被打上tag的任务和或者忽略被打上tag的任务

ansible-playbook playbook

6 ansible roles

Roles 是 Playbook的集合 便于复用某些 play以及甚至是task

6.1 Roles一览

在ansible 中, roles 通过文件的组织结构来展现。

高手都是建议 编排Roles的正确姿势是:

- 最优解
 - 从 galaxy 下载 role
 - 修改 roles
 - 使用
- 次优解
 - 手写 yaml & playbook

个人感想, 别人写的 roles 好像就是让你看不懂。不仅没有注释, 还没有好的 markdown

题外话:

看下面的例子想到了两个词:

抽象本质上是对某个属性的限制条件进行**泛化**以便更加形成具有一般性的概念

高内聚、松耦合, 模块化, 组件化了

如: [nginxinc](#)

```
ansible-galaxy install nginxinc.nginx
```

```
tree /root/.ansible/roles/nginxinc.nginx/
```

```
/root/.ansible/roles/nginxinc.nginx/
├── CHANGELOG.md
├── CODE_OF_CONDUCT.md
├── CONTRIBUTING.md
├── defaults
│   └── main
│       ├── amplify.yml
│       ├── bsd.yml
│       ├── logrotate.yml
│       ├── main.yml
│       ├── selinux.yml
│       └── systemd.yml
├── files
│   ├── license
│   └── services
│       ├── nginx.conf.upstart
│       ├── nginx.openrc
│       ├── nginx.override.conf
│       ├── nginx.systemd
│       ├── nginx.sysvinit
│       └── nginx.upstart
├── handlers
│   └── main.yml
├── LICENSE
├── meta
│   └── main.yml
├── molecule
│   ├── common
│   │   ├── Dockerfile.j2
│   │   └── playbooks
```

```
| | | | └─ default_converge.yml
| | | | └─ default_verify.yml
| | | | └─ module_converge.yml
| | | | └─ module_verify.yml
| | | | └─ plus_converge.yml
| | | | └─ plus_prepare.yml
| | | | └─ plus_verify.yml
| | | | └─ source_converge.yml
| | | | └─ source_verify.yml
| | └─ default
| | | └─ molecule.yml
| └─ default_alpine
| | └─ molecule.yml
└─ default_centos
| | └─ molecule.yml
└─ module
| | └─ molecule.yml
└─ module_alpine
| | └─ molecule.yml
└─ module_centos
| | └─ molecule.yml
└─ plus
| | └─ molecule.yml
└─ plus_alpine
| | └─ molecule.yml
└─ plus_centos
| | └─ molecule.yml
└─ source
| | └─ molecule.yml
└─ source_alpine
| | └─ molecule.yml
└─ source_centos
| | └─ molecule.yml
└─ README.md
└─ tasks
| └─ amplify
| | └─ install-amplify.yml
| | └─ setup-debian.yml
| | └─ setup-redhat.yml
| └─ config
| | └─ debug-output.yml
| | └─ modify-systemd.yml
| | └─ setup-logrotate.yml
| └─ keys
| | └─ setup-keys.yml
└─ main.yml
└─ modules
| | └─ install-modules.yml
└─ opensource
| | └─ install-alpine.yml
| | └─ install-bsd.yml
| | └─ install-debian.yml
| | └─ install-oss.yml
| | └─ install-redhat.yml
| | └─ install-source.yml
| | └─ install-suse.yml
└─ plus
| | └─ install-alpine.yml
```

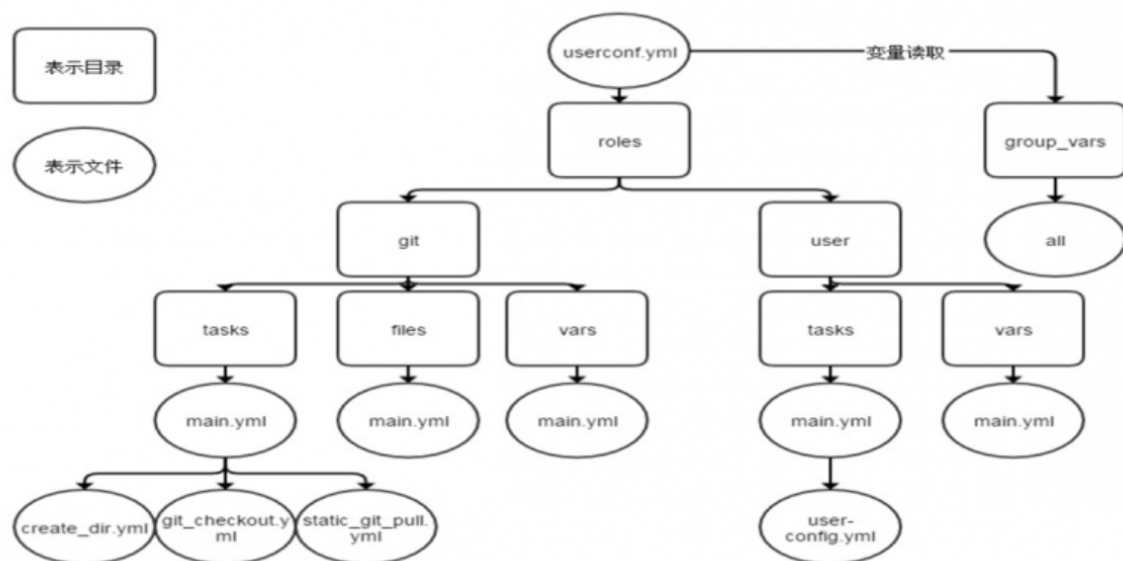
```

| | | └─ install-debian.yml
| | | └─ install-freebsd.yml
| | | └─ install-redhat.yml
| | | └─ install-suse.yml
| | | └─ remove-license.yml
| | | └─ setup-license.yml
| | └─ prerequisites
| |   └─ install-dependencies.yml
| |   └─ prerequisites.yml
| |   └─ setup-selinux.yml
└─ templates
  └─ logrotate
  |   └─ nginx.j2
  └─ selinux
  |   └─ nginx-plus-module.te.j2
  └─ services
  |   └─ nginx.service.override.conf.j2
└─ vars
  └─ main.yml

```

35 directories, 71 files

自己编写的 Roles 一般是这样子



比如应用发布就是如此

roles目录结构:

```

└─playbook.yml
└─roles/
  └─main.yml
└─project/
  └─main.yml
└─tasks/
  └─main.yml
└─files/
  └─main.yml
└─vars/

```

```

|   └─main.yml
├─templates/
|   └─main.yml
├─handlers/
|   └─main.yml
├─default/
|   └─main.yml
├─meta/
|   └─main.yml

```

Roles各目录作用

roles/project/ :项目名称,有以下子目录

- files/ : 存放由copy或script模块等调用的文件
- templates/: template模块查找所需要模板文件的目录
- tasks/: 定义task,role的基本元素, 至少应该包含一个名为main.yml的文件; 其它的文件需要在此文件中通过include进行包含
- handlers/: 至少应该包含一个名为main.yml的文件; 其它的文件需要在此文件中通过include进行包含
- vars/: 定义变量, 至少应该包含一个名为main.yml的文件; 其它的文件需要在此文件中通过include进行包含
- meta/: 定义当前角色的特殊设定及其依赖关系,至少应该包含一个名为main.yml的文件, 其它文件需在此文件中通过include进行包含
- default/: 设定默认变量时使用此目录中的main.yml文件, 比vars的优先级低

6.1 创建Role

创建role的步骤

- 创建以roles命名的目录
- 在roles目录中分别创建以各角色名称命名的目录, 如webserver等
- 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
- 在playbook文件中, 调用各角色

针对大型项目使用Roles进行编排

范例: roles的目录结构

```

nginx-role.yml
roles/
├─ nginx
│   ├── files
│   │   └─ main.yml
│   ├── tasks
│   │   ├── groupadd.yml
│   │   ├── install.yml
│   │   ├── main.yml
│   │   ├── restart.yml
│   │   └─ useradd.yml
│   └─ vars
│       └─ main.yml

```

6.2 Playbook 执行角色方法

常见的有三种

6.2.1 朴实无华的每个 role依次执行一遍

```
---
- hosts: webservs
  remote_user: root
  roles:
    - mysql
    - memcached
    - nginx
```

6.2.2 键role用于指定角色名称，后续的k/v用于传递变量给角色

```
---
- hosts: all
  remote_user: root
  roles:
    - mysql
    - { role: nginx, username: nginx }
```

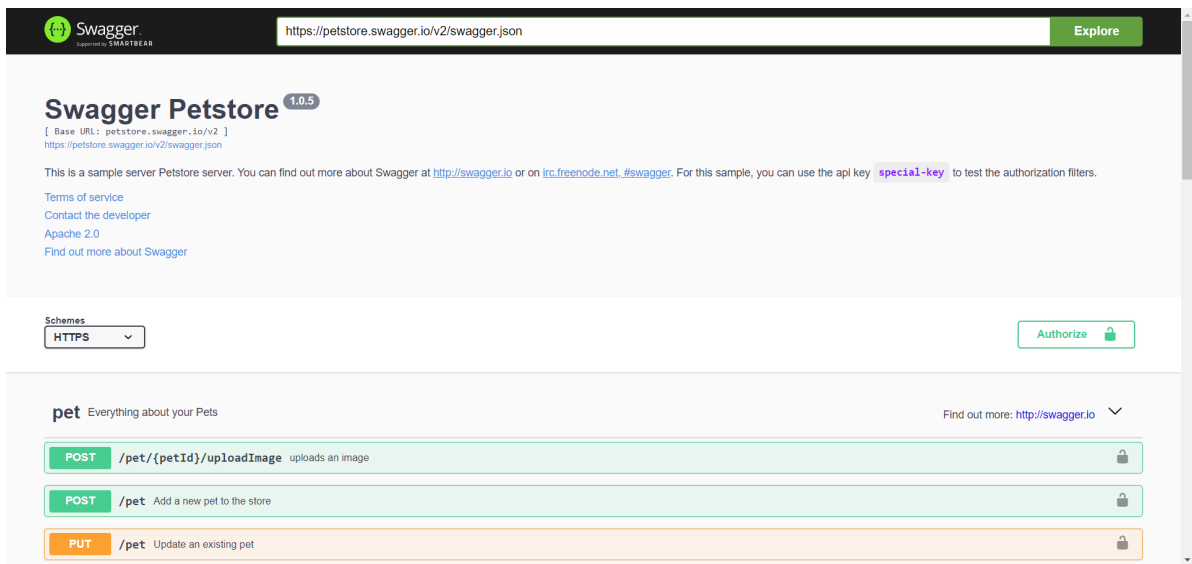
6.2.3 roles 中 tags 使用

```
#nginx-role.yml
---
- hosts: webservs
  remote_user: root
  roles:
    - { role: nginx ,tags: [ 'nginx', 'web' ] ,when:
      ansible_distribution_major_version == "6" }
    - { role: httpd ,tags: [ 'httpd', 'web' ] }
    - { role: mysql ,tags: [ 'mysql', 'db' ] }
    - { role: mariadb ,tags: [ 'mariadb', 'db' ] }

ansible-playbook --tags="nginx,httpd,mysql" nginx-role.yml
```

```
ansible-playbook playbook_07.yml
```

7 实操目标



Role repo: https://github.com/Sirius0301/hello_ansible_swagger.git

project repo: https://github.com/Sirius0301/swagger_static_test.git

8 奖励

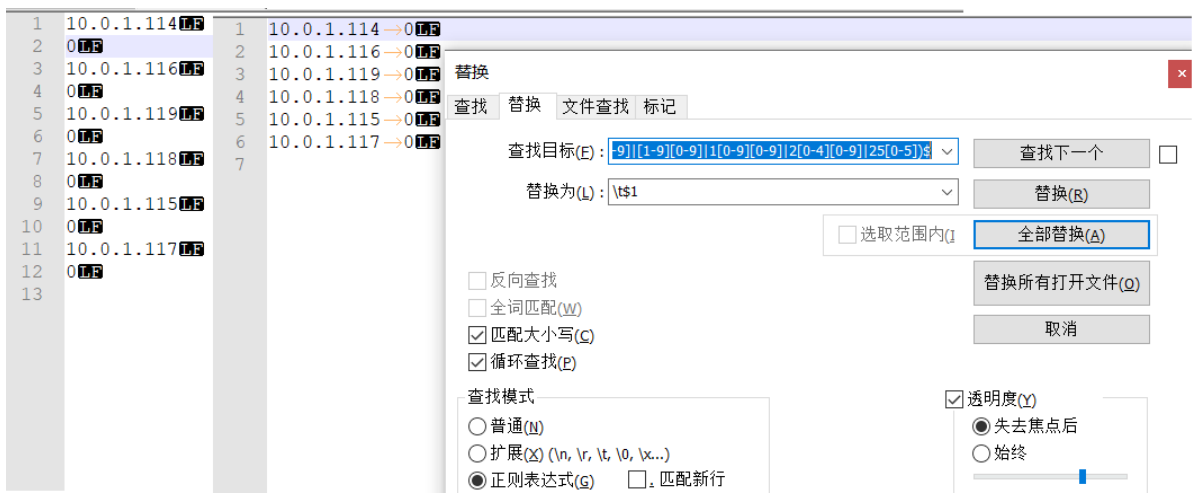
```
ansible all -m shell -a "history | wc -l" > rank.txt
```

#利用xftp copy rank.txt 到本地

#使用notepad 编辑， 正则表达式替换

```
\n^([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])$
```

```
\t$1
```



导入 Excel 排序，则可以挑出最大值

9 参考资料

视频资料：

1. <https://www.bilibili.com/video/BV18t411f7CN>

书籍资料：

1. Ansible--快速入门: <http://ansible.com.cn/index.html>
2. Ansible 指导手册: https://github.com/Sirius0301/hello_ansible_public/blob/master/Ansible_book.pdf

文档资料:

1. Ansible--快速入门: <https://www.cnblogs.com/yanjieli/p/10969089.html>
 2. Ansible--Module: <https://www.cnblogs.com/yanjieli/p/10969143.html>
 3. Ansible--Ansible之Playbook: <https://www.cnblogs.com/yanjieli/p/10969299.html>
 4. Ansible-- 运维派详解: <http://www.yunweipai.com/34663.html>
 5. Ansible--Ansible之Roles: <https://www.cnblogs.com/yanjieli/p/10971862.html>
 6. Ansible项目实战Inmp: <https://www.cnblogs.com/yanjieli/p/10980057.html>
 7. ansible服务部署与使用: <https://www.cnblogs.com/clsn/p/7743792.html>
 8. ansible 自动化运维详解 <https://www.cnblogs.com/keerya/p/7987886.html>
 9. 一步到位玩儿透Ansible <https://www.cnblogs.com/f-ck-need-u/p/7576137.html#ansible>
 10. Static Web With Nginx: <https://www.redhat.com/sysadmin/deploying-static-website-ansible>
-

End