nsd1902_devops_day04

ansible模块练习

1. 环境变量

```
# export ANSIBLE_LIBRARY=/opt/myansible_lib/
```

2. 编写模块

```
# vim /opt/myansible_lib/download.py
#!/usr/bin/env python
# -*- coding: utf8 -*-
'用于下载指定文件'

from ansible.module_utils.basic import AnsibleModule
import wget

def main():
    module = AnsibleModule(
        argument_spec=dict(
            url=dict(required=True, type='str'),
            path=dict(required=True, type='str')
        )
    )
    wget.download(module.params['url'], module.params['path'])
    module.exit_json(changed=True)

if __name__ == '__main__':
    main()
```

> 注意：远程主机只有python2，不是python3。python2默认字符采用的是ASCII码，而不是unicode，所以python2的文本中不能写汉字。如果需要添加汉字，需要在文件中指定字符集。

3. 在远程主机上安装wget。因为ansible准备好的模块，需要在远程主机上执行。

```
# 在本机上下载wget
# pip3 download wget --trusted-host pypi.douban.com

# 在远程主机上安装wget
# ansible webservers -m copy -a "src=files/wget-3.2.zip dest=/root"
# ansible webservers -m shell -a "cd /root; unzip wget-3.2.zip"
# ansible webservers -m shell -a "cd /root/wget-3.2/; python setup.py install"
```

4. 通过自己编写的模块，下载文件

```
# ansible webservers -m download -a "url=http://192.168.4.254/zabbix.png
path=/tmp/zabbix.png"
```

git

git基本应用

《pro git》https://down.51cto.com/data/273438

安装

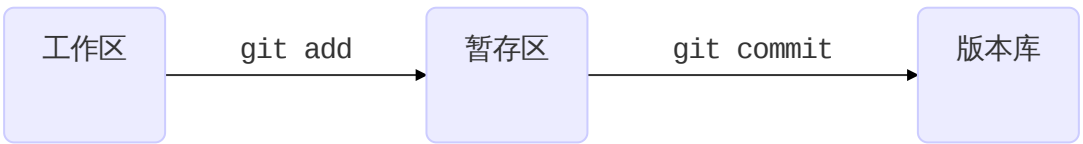
```
[root@node5 ~]# yum install -y git
```

配置基础信息

```
[root@node5 ~]# git config --global user.name "Mr.zzg"
[root@node5 ~]# git config --global user.email "zzg@tedu.cn"
[root@node5 ~]# git config --global core.editor vim
[root@node5 ~]# git config --list
[root@node5 ~]# cat ~/.gitconfig
```

git的重要工作区域

- 工作区：编写代码的项目目录
- 暂存区：工作区与版本库之间的缓冲地带
- 版本库：在工作区中有一个名为.git的目录，它是保存文件的目录



git应用流程

1. 创建版本库

```
# 方法一：还没有项目目录前
[root@node5 ~]# git init pro1
初始化空的 Git 版本库于 /root/pro1/.git/
[root@node5 ~]# ls -A pro1/
.git

# 方法二：已有项目目录
[root@node5 ~]# mkdir pro2
```

```
[root@node5 ~]# echo '<h1>my web site</h1>' > pro2/index.html
[root@node5 ~]# cd pro2/
[root@node5 pro2]# git init .
初始化空的 Git 版本库于 /root/pro2/.git/
[root@node5 pro2]# ls -A
.git  index.html
```

2. 添加文件到暂存区

```
[root@node5 pro2]# git status   # 查看状态
# 位于分支 master
#
# 初始提交
#
# 未跟踪的文件:
#   (使用 "git add <file>..." 以包含要提交的内容)
#
#   index.html
提交为空，但是存在尚未跟踪的文件（使用 "git add" 建立跟踪）
[root@node5 pro2]# git status -s
?? index.html
[root@node5 pro2]# git add index.html   # 提交到暂存区
[root@node5 pro2]# git status
# 位于分支 master
#
# 初始提交
#
# 要提交的变更：
#   (使用 "git rm --cached <file>..." 撤出暂存区)
#
#   新文件：    index.html
#
[root@node5 pro2]# git status
# 位于分支 master
#
# 初始提交
#
# 要提交的变更：
#   (使用 "git rm --cached <file>..." 撤出暂存区)
#
#   新文件：    index.html
#
[root@node5 pro2]# git status -s
A  index.html
```

3. 提交文件到版本库

```
[root@node5 pro2]# git commit -m "add new file index.html"
[root@node5 pro2]# git status
# 位于分支 master
无文件要提交，干净的工作区
[root@node5 pro2]# git status -s
```

4. 将文件从暂存区中撤出

```
[root@node5 pro2]# cp /etc/hosts .
[root@node5 pro2]# git add .   # 将当前工作区中的所有文件提交到暂存区
[root@node5 pro2]# git status
# 位于分支 master
# 要提交的变更：
#   （使用 "git reset HEAD <file>..." 撤出暂存区）
#
#   新文件：    hosts
#
[root@node5 pro2]# git reset HEAD hosts
[root@node5 pro2]# git status
# 位于分支 master
# 未跟踪的文件:
#   （使用 "git add <file>..." 以包含要提交的内容）
#
#   hosts
提交为空，但是存在尚未跟踪的文件（使用 "git add" 建立跟踪）
```

5. 删除文件

```
# 将尚未提交到版本库的文件提交
[root@node5 pro2]# git add .
[root@node5 pro2]# git commit -m "add hosts"

# 删除hosts
[root@node5 pro2]# rm -f hosts
[root@node5 pro2]# git status
# 位于分支 master
# 尚未暂存以备提交的变更：
#   （使用 "git add/rm <file>..." 更新要提交的内容）
#   （使用 "git checkout -- <file>..." 丢弃工作区的改动）
#
#   删除：      hosts
#
修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）
[root@node5 pro2]# git rm hosts
rm 'hosts'
[root@node5 pro2]# git status
# 位于分支 master
# 要提交的变更：
#   （使用 "git reset HEAD <file>..." 撤出暂存区）
#
#   删除：      hosts
#
[root@node5 pro2]# git commit -m "delete hosts"
[root@node5 pro2]# git status
# 位于分支 master
无文件要提交，干净的工作区

# 继续删除index.html
```

```
[root@node5 pro2]# git rm index.html
rm 'index.html'
[root@node5 pro2]# git status
# 位于分支 master
# 要提交的变更：
#   （使用 "git reset HEAD <file>..." 撤出暂存区）
#
#   删除：      index.html
#
[root@node5 pro2]# git reset HEAD index.html
重置后撤出暂存区的变更：
D   index.html
[root@node5 pro2]# git status
# 位于分支 master
# 尚未暂存以备提交的变更：
#   （使用 "git add/rm <file>..." 更新要提交的内容）
#   （使用 "git checkout -- <file>..." 丢弃工作区的改动）
#
#   删除：      index.html
#
修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）
[root@node5 pro2]# git  checkout -- index.html
[root@node5 pro2]# ls
index.html
[root@node5 pro2]# git status
# 位于分支 master
无文件要提交，干净的工作区
```

6. 移动、复制

```
[root@node5 pro2]# mkdir abc
[root@node5 pro2]# cp /etc/passwd .
[root@node5 pro2]# cp /etc/shadow .
[root@node5 pro2]# ls
abc  index.html  passwd  shadow

# 提交文件到版本库
[root@node5 pro2]# git add .
[root@node5 pro2]# git commit -m 'add new files'
[root@node5 pro2]# git status
# 位于分支 master
无文件要提交，干净的工作区

# 将passwd移动到abc中，改名为mima
[root@node5 pro2]# git mv passwd abc/mima
[root@node5 pro2]# git status
# 位于分支 master
# 要提交的变更：
#   （使用 "git reset HEAD <file>..." 撤出暂存区）
#
#   重命名：    passwd -> abc/mima
#
[root@node5 pro2]# git commit -m "mv passwd -> abc/mima"
```

```
[root@node5 pro2]# git status
# 位于分支 master
无文件要提交，干净的工作区

# 将shadow拷贝到abc
[root@node5 pro2]# cp shadow abc
[root@node5 pro2]# git add .
[root@node5 pro2]# git commit -m "cp shadow abc/"
```

分支管理

- git默认有一个名为master的分支
- git默认情况下，名为HEAD的指针指向master

返回到之前的某一次commit：

```
# 查看以前的所有提交
[root@node5 pro2]# git log
# 查看当前状态，务必保证当前状态是干净的
[root@node5 pro2]# git status
# 位于分支 master
无文件要提交，干净的工作区

# 回到add hosts这个时候的状态
[root@node5 pro2]# git checkout 9f1054366641074f386ae4348a8fa920e575c43f
[root@node5 pro2]# ls
hosts  index.html

# 回到最新的状态
[root@node5 pro2]# git checkout master
[root@node5 pro2]# ls
abc  index.html  shadow
```

创建新的分支

在编写程序时，可以在某一个提交点创建新的分支，在分支内编写代码。当分支代码的功能编写完后，再把它汇入主干。

```
# 查看分支
[root@node5 pro2]# git branch
* master

# 创建分支b1
[root@node5 pro2]# git branch b1
# 查看分支，分支名前有*号的，表示当前分支
[root@node5 pro2]# git branch
  b1
* master
# 在当前分支中，新增加文件
[root@node5 pro2]# cp /etc/redhat-release .
[root@node5 pro2]# git add .
[root@node5 pro2]# git commit -m "add redhat-release"
```

```
# 切换到b1分支
[root@node5 pro2]# git checkout b1
切换到分支 'b1'
[root@node5 pro2]# git branch
* b1
  master
[root@node5 pro2]# ls
abc  index.html  shadow
# 在b1分支中，新建文件
[root@node5 pro2]# echo 'hello world' > hello.txt
[root@node5 pro2]# ls
abc  hello.txt  index.html  shadow
[root@node5 pro2]# git add .
[root@node5 pro2]# git commit -m "add hello.txt"

# 当b1中的代码已经完成了，就可以将分支汇入主干
# 切换到master分支，并汇入
[root@node5 pro2]# git checkout master
切换到分支 'master'
[root@node5 pro2]# ls
abc  index.html  redhat-release  shadow
[root@node5 pro2]# git merge b1
[root@node5 pro2]# ls
abc  hello.txt  index.html  redhat-release  shadow

# 删除分支
[root@node5 pro2]# git branch -d b1
已删除分支 b1（曾为 8853c68）。
[root@node5 pro2]# git branch
* master
```

gitlab服务器

准备虚拟机

- 准备一台内存4G以上的虚拟机
- 安装docker
- 将gitlab-zh.tar镜像拷贝到虚拟机

配置docker

- 安装
- 导入镜像

```
[root@node6 images]# docker load < gitlab_zh.tar
```

- 修改虚拟机ssh端口为2022，因为22端口需要留给容器用

```
[root@node6 images]# vim /etc/ssh/sshd_config
Port 2022
[root@node6 images]# systemctl restart sshd
# 重新使用2022端口远程连接服务器
[root@room8pc16 nsd2019]# ssh -p2022 node6
```

- 启动容器

```
[root@node6 ~]# docker run -d -h gitlab --name gitlab -p 443:443 -p 80:80 -p 22:22 --
restart always -v /srv/gitlab/config:/etc/gitlab -v /srv/gitlab/logs:/var/log/gitlab -v
/srv/gitlab/data:/var/opt/gitlab gitlab_zh:latest
[root@node6 ~]# docker ps   # 查看状态
# 当状态中，显示healthy时，容器才能正常使用
```

配置gitlab

- 访问http://gitlab服务器
- 第一次访问需要为root用户设置密码，密码要求8位以上，复杂
- gitlab中重要的概念
  - 组group：可以为一个团队创建一个组
  - 成员member：创建用户，将用户加入到组中，成为组的成员
  - 项目project：一个团队、一个用户都可以开发软件项目，软件项目不只一个，为每一个开发项目，在gitlab上创建一个project与之对应。

创建名为devops的组，组类型为公开。

创建名为pro2的项目，属于devops组，类型公开。

创建用户。创建用户时，不能为用户指定密码，但是修改用户资料时，可以设置密码。

授权，授权新建的用户是pro2项止的主程序员。

用户配置

新用户不用配置任何选项，已经可以通过http协议进行代码的上传。但是，通过http上传代码，每次执行操作时，都需要填写用户名、密码。

配置ssh免密上传代码：

- 生成ssh密钥

```
[root@node5 pro2]# ssh-keygen -t rsa -C "zzg@tedu.cn" -b 4096
```

- 点击右上角用户“设置”
- 点击左侧工具栏的“ssh密钥”，将公钥拷贝到“密钥”文本框中

```
[root@node5 pro2]# cat ~/.ssh/id_rsa.pub
```

上传代码

```
[root@node5 pro2]# cd ~/pro2/
[root@node5 pro2]# git remote rename origin old-origin
# 以下错误可忽略
error: 不能重命名配置小节 'remote.origin' 到 'remote.old-origin'
[root@node5 pro2]# git remote add origin git@192.168.4.6:devops/pro2.git
[root@node5 pro2]# git push -u origin --all

# 后续上传代码的过程
[root@node5 pro2]# cp /etc/bashrc .
[root@node5 pro2]# git add .
[root@node5 pro2]# git commit -m "add bashrc"
[root@node5 pro2]# git push
```

其他用户下载代码：

- 协议应该使用http
- 第一次下载的时候，用git clone
- 后续如果有更新，只要git pull即可

```
[root@room8pc16 ~]# cd /tmp/
[root@room8pc16 tmp]# git clone http://192.168.4.6/devops/pro2.git
[root@room8pc16 tmp]# cd pro2/
[root@room8pc16 pro2]# ls
abc  bashrc  hello.txt  index.html  redhat-release  shadow
```