tedu_py0102

gitlab服务器

1. 克隆一台虚拟机，内存修改为4G
2. 配置IP地址、yum
3. 拷贝gitlab镜像和docker软件包到虚拟机

```
ftp://172.40.50.116/pub/phase5/docker/docker_pkgs
# cd docker_pkgs
# yum install -y *rpm
```

4. 安装docker并启动
5. 导入gitlab镜像

```
[root@node4 images]# docker load < gitlab_zh.tar
```

6. 把虚拟机ssh端口改为2022，因为22端口要留给gitlab使用

```
[root@node4 images]# vim /etc/ssh/sshd_config
Port 2022
[root@node4 images]# systemctl restart sshd
# 退出ssh再重新连接时，使用2022端口
[root@room8pc16 phase5]# ssh -p2022 192.168.4.4
```

7. 启动容器

```
[root@node4 ~]# 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
# 容器启动需要几分钟，当docker ps查看时，显示healty才算正常
```

gitlab中的关键概念

- 群组group：一个团队对应一个组
- 成员member：把用户加入到组中，成为组的成员
- 项目project：每个开发项目对应到gitlab中，成为一个项目

配置gitlab

- url: http://x.x.x.x/
- 第一次访问需要设置管理员root密码，8位，符合复杂度要求
- 用户名为root
- 点击创建组，创建名为devops的组，公开的组

- 任何时候都可以点击页面上的扳手图标，查看主要的操作
- 点击“创建用户”，创建一个用户Mrzhang。注意，创建用户的时候，不能设置密码。但是创建完用户后，可以点击“编辑”再设置密码。
- 点击页面上的扳手图标，点击创建项目。为组创建项目，因此项目路径要把root改为devops，最后项目名称随便填一个。可见等级选“公开”
- 创建好项目后，点击左侧工具栏中的齿轮图标->成员->选择要邀请的成员: Mrzhang -> 选择角色权限: 主程序员 -> 点击添加到项目

访问gitlab

当用户上传代码时，可以用HTTP协议或SSH协议。但是HTTP协议每次推代码时都需要填写gitlab的用户名和密码。用ssh可以实现免密。

实现ssh免密推送代码：

- 切换成普通用户登陆
- 点击右上角用户的设置 - > 左侧工具栏的ssh密钥 -> 把你自己电脑的ssh公钥粘贴到“密钥”文本框中
- 生成密钥

```
[root@node3 ~]# ssh-keygen -t rsa -C "Mrzhang@tedu.cn" -b 4096
[root@node3 ~]# cat ~/.ssh/id_rsa.pub
```

上传代码：

```
[root@node3 ~]# cd mypro2/
[root@node3 mypro2]# git remote rename origin old-origin
# 出现以下错误可忽略
error: 不能重命名配置小节 'remote.origin' 到 'remote.old-origin'
[root@node3 mypro2]# git remote add origin \
  git@192.168.4.4:devops/mypro2.git
[root@node3 mypro2]# git push -u origin --all
[root@node3 mypro2]# git push -u origin --tags


# 以后继续编写代码并上传
[root@node3 mypro2]# cp /etc/passwd .
[root@node3 mypro2]# git add .
[root@node3 mypro2]# git commit -m "add passwd"
[root@node3 mypro2]# git push    # 推到服务器
```

其他人下载代码：

```
[root@room8pc16 tmp]# git clone http://192.168.4.4/devops/mypro2.git
[root@room8pc16 tmp]# ls mypro2/
index.html  passwd  zhuji
```

```
# 列表解析：快速生成列表
>>> [5]
[5]
>>> [5 + 2]
[7]
>>> [5 + 2 for i in range(5)]   # 循环决定5+2计算几次
[7, 7, 7, 7, 7]
>>> [5 + i for i in range(1, 6)]  # 表达式可以用for中的变量
[6, 7, 8, 9, 10]
# 判断条件为真才把5+i的结果保存到列表
>>> [5 + i for i in range(1, 11) if i % 2 == 0]
[7, 9, 11, 13, 15]
```

文件

操作步骤：打开、读写、关闭

打开文件采用的模式：

- 读：r 文件不存在则报错，不能写
- 写：w 文件不存在则创建，存在则清空。不能读
- 追加：a 文件不存在则创建，存在则追加写入。
- 打开为bytes类型：b 如果打开非文本文件，必须以bytes类型打开

操作文本文件

读操作

```
[root@room8pc16 weekend1]# cp /etc/passwd /tmp/
>>> f = open('/tmp/passwd')   # 默认以r方式打开
>>> data = f.read()   # 默认读全部数据
>>> print(data)   # 可以看到文件的全部内容
>>> data = f.read()   # 因为已经读到尾部，再读已无数据
>>> data
''
>>> f.close()   # 关闭文件

>>> f = open('/tmp/passwd')
>>> f.read(4)   # 最多读4字节
'root'
>>> f.read(4)   # 继续向后读4字节
':x:0'
>>> f.readline()  # 从文件指针位置开始读一行
':0:root:/root:/bin/bash\n'
>>> f.readlines()  # 把剩余所有内容读到列表中
>>> f.close()

# 遍历文本文件更常用的方法是for循环
>>> f = open('/tmp/passwd')
>>> for line in f:
...     print(line, end='')
>>> f.close()
```

```
# 读非文本文件（文本文件也可以这样操作）
[root@room8pc16 weekend1]# cp /bin/ls /tmp/
>>> f = open('/tmp/ls', 'rb')
>>> f.read(10)
b'\x7fELF\x02\x01\x01\x00\x00\x00'   # b表示bytes类型，\x表示16进制
>>> f.close()
```

写操作

```
>>> f = open('/tmp/passwd', 'w')
[root@room8pc16 weekend1]# cat /tmp/passwd    # 已清空
>>> f.write('hello world!\n')   # \n是回车换行
13    # 表示写入了13个字节
>>> f.writelines(['2nd line.\n', '3rd line.\n'])
[root@room8pc16 weekend1]# cat /tmp/passwd   # 还是空的
>>> f.flush()  # 立即同步数据到硬盘
>>> f.write('the end.\n')
>>> f.close()   # 关闭文件也会写入硬盘
```

with语句

通过with语句打开文件，with语句结束后，文件自动关闭

```
>>> with open('/tmp/passwd') as f:
...     f.readline()
...
'hello world!\n'
>>> f.readline()   # 报错，因为文件已经关闭
```

seek语句

seek用于移动文件指针

```
>>> f = open('/tmp/passwd', 'rb')
>>> f.seek(6, 0)   # 指针移动到从开头（0）向右偏移6字节处
>>> f.read(5)
b'world'
>>> f.seek(6, 1)    # 指针从当前位置（1）向右移动6字节
17
>>> f.seek(-5, 2)   # 指针从末尾（2）向左移动5个字节
>>> f.read()
b'end.\n'
>>> f.close()
```

函数

相当于把一组功能代码起个名，方便以后调用。

模块

一个python程序文件就是一个模块。

```
# vim star.py
'''
star

包括一个全局变量和一个函数
'''
hi = 'hello world'
def pstar(n=30):
    '用于打印一行星号'
    print('*' * n)

# python3
>>> import star
>>> help(star)
>>> star.hi
'hello world'
>>> star.pstar()
******************************
```

导入(import)模块时，模块中的代码会执行(load)一遍。但是不管导入多少回，都以第一次为准。

导入模块的方法：

```
>>> import sys
>>> from random import randint, choice
>>> randint(1, 10)
8
>>> choice('abcd')
'a'

# 不常用的导入方法
>>> import sys, time, os   # 多个模块一行导入
>>> import getpass as gp   # 导入模块时给它起个别名
>>> gp.getpass()
Password:
```

模块的特殊属性

每个模块都有一个名为__name__的变量，当模块直接运行（python3 xxx.py）时，它的值是__main__；当模块间接运行（其他文件导入该模块）时，它的值是模块名。

```
[root@room8pc16 py0102]# cat foo.py
print(__name__)
[root@room8pc16 py0102]# cat bar.py
import foo
[root@room8pc16 py0102]# python3 foo.py
__main__
[root@room8pc16 py0102]# python3 bar.py
foo
```

shutil模块

```
>>> import shutil
>>> f1 = open('/bin/ls', 'rb')
>>> f2 = open('/tmp/list2', 'wb')
>>> shutil.copyfileobj(f1, f2)
>>> f1.close()
>>> f2.close()

# 拷贝文件内容
>>> shutil.copyfile('/bin/ls', '/tmp/list3')
# 拷贝文件内容和权限
>>> shutil.copy('/bin/ls', '/tmp/list4')
# copy2相当于cp -p
>>> shutil.copy2('/bin/ls', '/tmp/list5')
>>> shutil.copytree('/etc/security', '/tmp/anquan')   # cp -r
>>> shutil.move('/tmp/anquan', '/var/tmp/anquan')  # mv
>>> shutil.rmtree('/var/tmp/anquan')   # rm -rf
>>> shutil.chown('/tmp/list', user='zhangsan', group='zhangsan')
```