Git：分布式版本控制工具

1. 安装

# yum install -y git

1. 提交代码需要配置个人信息

[root@room8pc16 python]# git config --global user.name "zhangzhg"

[root@room8pc16 python]# git config --global user.email "zhangzg@tedu.cn"

3、设置编写代码说明的编辑器是vim

[root@room8pc16 python]# git config --global core.editor vim

1. 查看

[root@room8pc16 python]# git config --list

[root@room8pc16 python]# cat ~/.gitconfig

1. 创建工作区

[root@room8pc16 ~]# mkdir mycode

1. 初始化版本库

[root@room8pc16 ~]# cd mycode

[root@room8pc16 mycode]# git init .

[root@room8pc16 mycode]# ls -a

1. 编写程序文件

[root@room8pc16 mycode]# echo 'hello world' > hi.txt

[root@room8pc16 mycode]# git status 查看状态

1. 添加跟踪文件（所有文件）到版本库

[root@room8pc16 mycode]# git add .

[root@room8pc16 mycode]# git status

1. 提交文件到版本库

[root@room8pc16 mycode]# git commit -m "add hi.txt"

[root@room8pc16 mycode]# git status

1. 修改hi.txt

[root@room8pc16 mycode]# echo "new line" >> hi.txt

[root@room8pc16 mycode]# git add .

[root@room8pc16 mycode]# git commit -m "modify hi.txt"

1. 恢复hi.txt到以前版本

[root@room8pc16 mycode]# git log

查看第一次提交的ID号，它的显示如下：

commit 48c488c8efb45b2c31afa225c0d7ad281ecb6b11

Author: MrZhangzhg <zhangzg@tedu.cn>

Date: Mon Jul 16 11:23:39 2018 +0800

add hi.txt

[root@room8pc16 mycode]# git checkout 48c488c8efb45b2c31afa225c0d7ad281ecb6b11

1. 如果有误加入到版本库的文件，可以查到它，并删除

[root@room8pc16 mycode]# git ls-files 查看版本库中的文件

[root@room8pc16 mycode]# git rm hi.txt

[root@room8pc16 mycode]# git commit -m "delete hi.txt"

搭建gitlab服务器

1. 创建一台虚拟机，内存加到4G

node1.tedu.cn 192.168.4.1

1. 把docker程序和镜像拷贝到虚拟机中

[root@room8pc16 phase5]# scp -r docker 192.168.4.1:/root

3、安装docker

[root@node1 ~]# rpm -ihv docker/docker\_pkgs/\*rpm

1. 启动服务

[root@node1 ~]# systemctl start docker

[root@node1 ~]# systemctl enable docker

5、导入镜像

[root@node1 ~]# docker load < docker/images/gitlab\_zh.tar

1. 为了方便gitlab容器的运行，将node1的ssh服务切换成2222端口

[root@node1 ~]# vim /etc/ssh/sshd\_config

Port 2222

[root@node1 ~]# systemctl restart sshd

7、重新登陆到node1

[root@room8pc16 nsd2018]# ssh node1 -p 2222

8、启动新容器。将将容器的443、80、22端口发布出去。当容器意外停止的时候，将其重启。再将容器的配置目录、日志目录、数据目录映射到本/srv/gitlab目录。

[root@node1 ~]# 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/log:/var/log/gitlab -v /srv/gitlab/data gitlab\_zh:latest

1. 配置gitlab
2. 访问http://192.168.4.1，第一次访问需要设置密码，密码必须8位以上。如1234.com
3. 登陆时，用户名是root，密码是1234.com
4. 创建群组，群组路径和名字填写devops，类型为公开
5. 创建群组后，右下角有创建项目，点击以创建项目
6. 拉取项目测试

[root@room8pc16 tmp]# cd /tmp/

[root@room8pc16 tmp]# git clone http://192.1689.4.1/devops/core\_py.git

以下是三种情况的使用说明：第一是先在gitlab上创建项目，然后clone到本地，最后在本地进入目录开始编写代码；第二种情况是本地已有一个目录，但是还没有加入到版本库管理；第三种情况是本地已有目录，并且已经通过git init初始化过版本库了



1. 创建用户、授权可以向项目中提交代码。

点击Web页面上方的扳手图标，点击新建用户。新建用户时不能设置密码，创建成功后，点击“编辑”可以设置。

1. root用户将新建的用户加入到群组中，并且设置新用户为“主程序员”，用户就可以上传代码
2. 在本地配置新用户，实现ssh上传代码
3. 在gitlab的web页面上注销root用户，用新用户登陆，首次登陆需要修改密码，新老密码可以一样。
4. 用户在本地生成ssh密钥

[root@node1 ~]# ssh-keygen -t rsa -C "zhangzg@tedu.cn" -b 4096

1. 查看密钥内容

[root@node1 ~]# cat /root/.ssh/id\_rsa.pub

1. 在gitlab页面上点击左侧边栏的ssh密钥，把第（3）步查看到的密钥内容粘贴进来
2. 本地上传代码测试

（1）创建本地版本库

[root@node1 ~]# mkdir myproject

[root@node1 ~]# cd myproject/

[root@node1 myproject]# cp /etc/hosts .

[root@node1 myproject]# git init

[root@node1 myproject]# git add .

[root@node1 myproject]# git commit -m "init myproject"

（2）上传代码

[root@node1 myproject]# git remote rename origin old-origin

如果出现以下错误，可以忽略

error: 不能重命名配置小节 'remote.origin' 到 'remote.old-origin'

[root@node1 myproject]# git remote add origin git@192.168.4.1:devops/core\_py.git

[root@node1 myproject]# git push -u origin --all

1. 如果有代码的修改，只要git add / git commit /git push即可

[root@node1 myproject]# cp /etc/passwd .

[root@node1 myproject]# git add .

[root@node1 myproject]# git commit -m "add new file passwd"

[root@node1 myproject]# git push