Git与Python

1. Git（分布式的版本控制工具）

1.git的基本使用：

（1）安装：

[root@room9pc01 ~]# yum -y install git

（2）提交代码需要配置个人信息（用户名和用户邮箱）：

[root@room9pc01 ~]# git config --global user.name "Coldsama"

[root@room9pc01 ~]# git config --global user.email \ "Coldsama@163.com"

（3）设置编写代码说明的编辑器是vim

[root@room9pc01 ~]# git config --global core.editor vim

[root@room9pc01 ~]# git config --list //查看用户信息

user.name=Coldsama

user.email=Coldsama1213@163.com

core.editor=vim

（4）创建工作区并初始化版本库

[root@room9pc01 ~]# mkdir mycode

[root@room9pc01 ~]# cd mycode/

[root@room9pc01 mycode]# git init .

（5）编写程序文件及提交

[root@room9pc01 mycode]# echo 'hello world' > hello.py

[root@room9pc01 mycode]# git status //查看状态

# 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）

# hello.py

......

[root@room9pc01 mycode]# git add hello.py //将程序添加进缓存区

[root@room9pc01 mycode]# git status

# 位于分支 master

......

要变更的条件

# 新文件： hello.py

[root@room9pc01 mycode]# git commit -m "add hello.py" //提交代

码版本库中

注：-m：添加本次修改的注释。

若缓存区有多个程序文件，可使用-a选项全部提交。

（6）修改程序文件

[root@room9pc01 mycode]# echo 'hello sword art online' >> hello.py

[root@room9pc01 mycode]# git status

# 头指针分离于 040dffb

# 尚未暂存以备提交的变更：

# （使用 "git add <file>..." 更新要提交的内容）

......

[root@room9pc01 mycode]# git add .

[root@room9pc01 mycode]# git status //注意看前后两次status的区别

# 头指针分离于 040dffb

# 要提交的变更：

# （使用 "git reset HEAD <file>..." 撤出暂存区）

....

[root@room9pc01 mycode]# cat hello.py

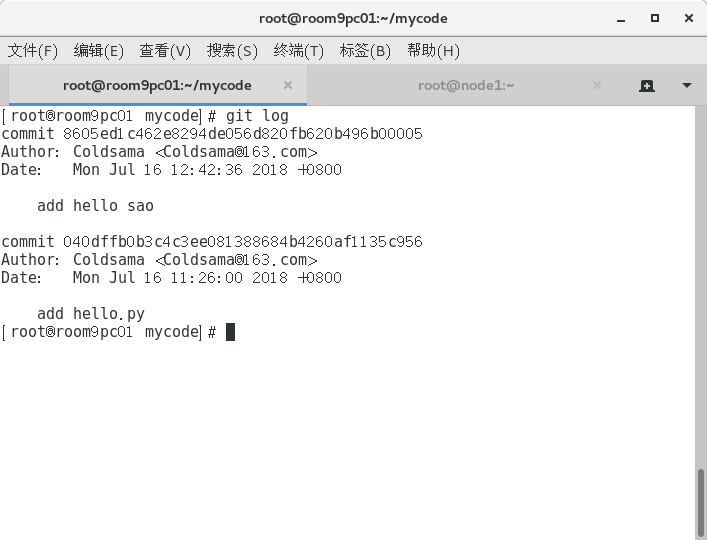
hello world

hello sword art online

（7）如何利用git恢复到以前的版本：

[root@room9pc01 mycode]# git log //根据修改的注释找到想恢复的版本

复制其commit号



[root@room9pc01 mycode]# git checkout \

040dffb0b3c4c3ee081388684b4260af1135c956

[root@room9pc01 mycode]# cat hello.py

hello world

（8）如何删除版本库中没用的文件：

[root@room9pc01 mycode]# git ls-files //查看版本库中的文件

hello.py

[root@room9pc01 mycode]# git rm hello.py //从版本库中移除指定文件

rm 'hello.py'

[root@room9pc01 mycode]# git commit -m 'delete hello.py' //移除注释

[root@room9pc01 mycode]# git ls-files //版本库中的文件已移除

[root@room9pc01 mycode]#

1. 搭建gitlab服务器

1.创建一台虚拟机，内存加到4G

2.把docker程序和镜像拷贝到虚拟机中并安装

[root@room9pc01 ~]# scp '/root/桌面/docker\_pkgs' [root@192.168.4.1:./](mailto:root@192.168.4.1:./)

[root@node1 ~]# cd docker\_pkgs/

[root@node1 docker\_pkgs]# ls

container-selinux-2.42-1.gitad8f0f7.el7.noarch.rpm

container-storage-setup-0.8.0-3.git1d27ecf.el7.noarch.rpm

docker-1.13.1-53.git774336d.el7.centos.x86\_64.rpm

docker-client-1.13.1-53.git774336d.el7.centos.x86\_64.rpm

docker-common-1.13.1-53.git774336d.el7.centos.x86\_64.rpm

oci-register-machine-0-6.git2b44233.el7.x86\_64.rpm

oci-systemd-hook-0.1.15-2.gitc04483d.el7.x86\_64.rpm

oci-umount-2.3.3-3.gite3c9055.el7.x86\_64.rpm

skopeo-containers-0.1.28-1.git0270e56.el7.x86\_64.rpm

[root@node1 docker\_pkgs]# rpm -ivh \*.rpm

3.启动服务

[root@node1 ~]# systemctl start docker

[root@node1 ~]# systemctl enable docker

[root@node1 ~]# docker ps

4.修改虚拟机的ssh服务端口为2222(因为docker的端口也是22，避免冲突)

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

Port 2222

[root@node1 ~]# systemctl restart sshd

[root@node1 ~]# exit

[root@room9pc01 ~]# ssh -X root@192.168.4.1 -p 2222

5.启动容器,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:/etcgitlab -v \ /srv/gitlab/log:/var/log/gitlab -v /srv/gitlab/data gitlab\_zh:latest

注：-d:后台运行容器；-h:指定容器的hostname；--name:容器名称；

-p:端口映射，格式是主机端口：容器端口；-v:挂载宿主机目录;

--restart always:自动重启容器。

[root@node1 ~]# ls /srv/gitlab

config log

6.配置gitlab服务器

（1）浏览器访问192.168.4.1，进入gitlab界面，修改密码为1234.com（默认修改的是root的密码）。

（2）登陆时，用户名是root，密码是1234.com

（3）创建群组

（4）创建新项目

（5）物理机拉取项目测试

[root@room9pc01 ~]# cd /tmp/

[root@room9pc01 tmp]# git clone http://192.168.4.1/devops/corepy.git

[root@room9pc01 tmp]# ls /tmp/

corepy wps-root

注：corepy是前面创建项目时对项目的命名。

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。

9.在本地配置新用户，实现ssh上传代码

（1）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。

（2）用户在本地生成ssh密钥

[root@room9pc01 ~]# ssh-keygen -t rsa -C ['Coldsama1213@163.com'](mailto:'Coldsama1213@163.com')

（3）查看密钥信息并复制

[root@room9pc01 ~]# cat .ssh/id\_rsa.pub //复制全部内容

（4）在gitlab页面上点击右上角个人头像，选设置，页面左侧边栏有SSH密钥的选项，将刚刚复制的密钥信息粘贴进文本框中。

1. 本地上传文件测试

（1）创建本地库

[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

（3）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

1.python ---->蟒蛇的意思，1991年发布第一个公开版本。

2.版本：目前所有系统默认安装python2.x版本，主流版本。

学习版本用的是python3.x版本。

3.特点：高级的数据结构，缩短开发时间与代码量、面向对象、可升级、可扩展、可移植性、易学易读、内存管理器。

4.