GIT

案例1:Git基本操作

1.1问题

本案例要求先快速搭建一台Git服务器.并测试该版本控制软件,要求如下:

\*安装Git软件

\*创建版本库

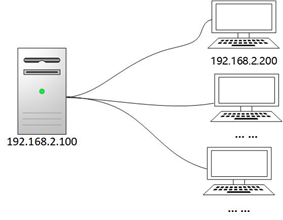
\*客户端克隆版本仓库到本地

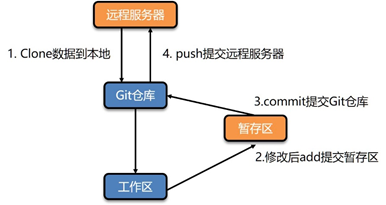
\*本地工作目录修改数据

\*提交本地修改到服务器

1.2方案

实验拓扑如图





1.3步骤

实现此案例需要按照如下步骤

步骤一:部署Git服务器(192.168.2.100为远程Git服务器)

[1]YUM安装Git软件

]#yum -y install git

]#git -version

[2]初始化空仓库

[root@web1 ~]# mkdir /var/git

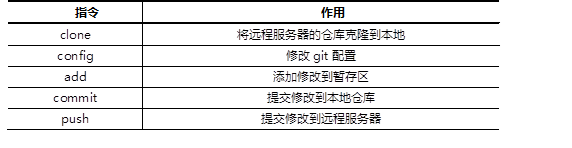
[root@web1 ~]# git init /var/git/project --bare

[root@web1 ~]# ls /var/git/project

config description HEAD hooks info objects refs

步骤二:客户端测试(192.168.2.200)作为客户端主机

使用Git常用的指令表



[1]clone克隆服务器仓库到本地

[root@web2 ~]# yum -y install git

[root@web2 ~]# git clone root@192.168.2.100:/var/git/project

[root@web2 ~]# cd project

[root@web2 ~]# ls

[2]修改git配置

[root@web2 project]# git config --global user.email "you@example.com"

[root@web2 project]# git config --global user.name "Your Name"

[root@web2 project]# cat ~/.gitconfig

[user]

    email = you@example.com

    name = Your Name

[3]本地工作区对数据进行增删该查(必须要先进入仓库在操作数据)

[root@web2 project]# echo "init date" > init.txt

[root@web2 project]# mkdir demo

[root@web2 project]# cp /etc/hosts demo

[4]查看仓库中数据的状态

]#git status

[5]将工作区的修改提交到暂存区

]#git add .

[6]将暂存区修改提交到本地仓库

[root@web2 project]# git commit -m "注释，可以为任意字符"

[root@web2 project]# git status

[7]将本地仓库中的数据推送到远程服务器(web2数据推送web1)

[root@web2 project]# git config --global push.default simple

[root@web2 project]# git push

root@192.168.2.100's password: 输入服务器root密码

[root@web2 project]# git status

[8]将服务器上的数据更新到本地（web1的数据更新到web2）。

备注：可能其他人也在修改数据并提交服务器，就会导致自己的本地数据为旧数据，使用pull就可以将服务器上新的数据更新到本地。

]#git pull

[9]查看版本日志

[root@web2 project]# git log //显示版本日志

[root@web2 project]# git log --pretty=oneline //一行显示版本日志

[root@web2 project]# git log --oneline //精简版本日志

[root@web2 project]# git reflog

备注:客户端也可以使用图形程序访问服务器.

windows需要安装

git和tortoiseGit

案例2:HEAD指针操作

2.1问题

沿用练习一,学习操作HEAD指针,具体要求如下:

\*查看Git版本信息

\*移动指针

\*通过移动HEAD指针恢复数据

2.2方案

HEAD指针是一个可以再任何分支和版本移动时的指针,通过移动指针我们可以将数据还原至任何版本.没做一次提交操作都会导致更新一个版本,HEAD指针也跟着自动移动

2.3步骤

步骤一:HEAD指针基本操作

[1]准备工作(多对数据仓库进行修改,提交操作.以产生多个版本).

[root@web2 project]# echo "new file" > new.txt

[root@web2 project]# git add .

[root@web2 project]# git commit -m "add new.txt"

[root@web2 project]# echo "first" >> new.txt

[root@web2 project]# git add .

[root@web2 project]# git commit -m "new.txt:first line"

[root@web2 project]# echo "second" >> new.txt

[root@web2 project]# git add .

[root@web2 project]# git commit -m "new.txt:second"

[root@web2 project]# echo "third" >> new.txt

[root@web2 project]# git add .

[root@web2 project]# git commit -m "new.txt:third"

[root@web2 project]# git push

[root@web2 project]# echo "123" > num.txt

[root@web2 project]# git add .

[root@web2 project]# git commit -m "num.txt:123"

[root@web2 project]# echo "456" > num.txt

[root@web2 project]# git add .

[root@web2 project]# git commit -m "num.txt:456"

[root@web2 project]# echo "789" > num.txt

[root@web2 project]# git add .

[root@web2 project]# git commit -m "num.txt:789"

[root@web2 project]# git push

[2]查看Git版本信息

[root@web2 project]# git reflog

[root@web2 project]# git log --oneline

04ddc0f num.txt:789

7bba57b num.txt:456

301c090 num.txt:123

b427164 new.txt:third

0584949 new.txt:second

ece2dfd new.txt:first line

e1112ac add new.txt

1a0d908 初始化

[3]移动HEAD指针,将数据还原到任意版本.

提示:当前HEAD指针为HEAD@{0}

[root@web2 project]# git reset --hard 301c0

[root@web2 project]# git reflog

301c090 HEAD@{0}: reset: moving to 301c0

04ddc0f HEAD@{1}: commit: num.txt:789

7bba57b HEAD@{2}: commit: num.txt:456

301c090 HEAD@{3}: commit: num.txt:123

b427164 HEAD@{5}: commit: new.txt:third

0584949 HEAD@{6}: commit: new.txt:second

ece2dfd HEAD@{7}: commit: new.txt:first line

e1112ac HEAD@{8}: commit: add new.txt

1a0d908 HEAD@{9}: commit (initial): 初始化

[root@web2 project]# cat num.txt            #查看文件是否为123

123

[root@web2 project]# git reset --hard 7bba57b

[root@web2 project]# cat num.txt        #查看文件是否为123，456

[root@web2 project]# git reflog          #查看指针移动历史

7bba57b HEAD@{0}: reset: moving to 7bba57b

301c090 HEAD@{1}: reset: moving to 301c0

… …

[root@web2 project]# git reset --hard 04ddc0f    #恢复num.txt的所有数据

[4]模拟误删后的数据还原操作

[root@web2 project]# git rm init.txt           #删 除文件

rm 'init.txt'

[root@web2 project]# git commit -m "delete init.txt"    #提交本地仓库

[root@web2 project]# git reflog           #查看版本历史

0dc2b76 HEAD@{0}: commit: delete init.txt

7bba57b HEAD@{0}: reset: moving to 7bba57b

301c090 HEAD@{1}: reset: moving to 301c0

… …

[root@web2 project]# git reset --hard 04ddc0f   #恢复数据

[root@web2 project]# ls

demo init.txt new.txt num.txt

案例3:Git分支操作

3.1问题

沿用练习二,学习操作GIT分支,集体要求如下

\*查看分支

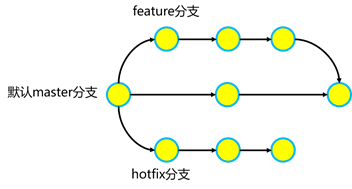
\*创建分支

\*切换分支

\*合并分支

\*解决分支冲突

3.2方案

Git支持按功能模块,时间,版本等标准创建分支,分支可以让开发分多条主线同时进行,每条主线互不影响,分支效果如图:

常见的分支规范如下:

常见的分支规范如下：

MASTER分支（MASTER是主分支，是代码的核心）。

DEVELOP分支（DEVELOP最新开发成果的分支）。

RELEASE分支（为发布新产品设置的分支）。

HOTFIX分支（为了修复软件BUG缺陷的分支）。

FEATURE分支（为开发新功能设置的分支）。

步骤一:查看并创建你分支

[1]查看当前分支

[root@web2 project]# git status

# On branch master

nothing to commit, working directory clean

[root@web2 project]# git branch -v

\* master 0dc2b76 delete init.txt

[2]创建分支

[root@web2 project]# git branch hotfix

[root@web2 project]# git branch feature

[root@web2 project]# git branch -v

feature 0dc2b76 delete init.txt

hotfix 0dc2b76 delete init.txt

\* master 0dc2b76 delete init.txt

步骤二:切换与合并分支

[1]切换分支

[root@web2 project]# git checkout hotfix

[root@web2 project]# git branch -v

feature 0dc2b76 delete init.txt

\* hotfix 0dc2b76 delete init.txt

master 0dc2b76 delete init.txt

[2]在新的分支上可以继续进行数据操作(增,删,改,cha)

[root@web2 project]# echo "fix a bug" >> new.txt

[root@web2 project]# git add .

[root@web2 project]# git commit -m "fix a bug"

[3]hotfix修改的数据合并到master分支。

注意，合并前必须要先切换到master分支，然后再执行merge命令。

[root@web2 project]# git checkout master

[root@web2 project]# cat new.txt        #默认master分支中没有hotfix分支中的数据

[root@web2 project]# git merge hotfix

Updating 0dc2b76..5b4a755

Fast-forward

new.txt | 1 ++

file changed, 1 insertions(+)

[4]将所有本地修改提交远程服务器

]#git push

步骤二:解决版本分支的冲突问题

[1]在不同分支中修改相同文件的相同行数据,模拟数据冲突

[root@web2 project]# git checkout hotfix

[root@web2 project]# echo "AAA" > a.txt

[root@web2 project]# git add .

[root@web2 project]# git commit -m "add a.txt by hotfix"

[root@web2 project]# git checkout master

[root@web2 project]# echo "BBB" > a.txt

[root@web2 project]# git add .

[root@web2 project]# git commit -m "add a.txt by master"

[root@web2 project]# git merge hotfix

自动合并 a.txt

冲突（添加/添加）：合并冲突于 a.txt

自动合并失败，修正冲突然后提交修正的结果。

[2]查看有冲突的文件内容,修改文件为最终版本的数据,解决冲突

[root@web2 project]# cat a.txt                #该文件中包含有冲突的内容

<<<<<<< HEAD

BBB

=======

AAA

>>>>>>> hotfix

[root@web2 project]# vim a.txt #修改该文件，为最终需要的数据，解决冲突

BBB

[root@web2 project]# git add .

[root@web2 project]# git commit -m "resolved"

总结:分支指针与HEAD指针的关系

\*创建分支的本质是在当前提交上创建一个可以移动的指针

\*如何判断当前分支呢?答案是根据HEAD这个特殊指针.

案例4.Git服务

4.1问题

沿用练习三,学习Git不同的服务器形式,客户端就可以使用不同的形式访问服务器.创建的服务器协议有SSH协议,Git协议,HTTP协议

步骤一:SSH协议服务器

[1]创建基于密码验证的SSH协议服务器(支持读写操作)

]#git init -bare /var/git/base\_ssh

Initialized empty Git repository in /var/git/base\_ssh/

[2]客户端访问方式(web2主机操作)

]#git clone [root@192.168.2.100:/var/git/base\_ssh](mailto:root@192.168.2.100:/var/git/base_ssh)

]#rm -rf base\_ssh

[3]客户端生成SSH秘钥,实现,免迷登录git服务器(web2主机操作)

[root@web2 ~]# ssh-keygen -f /root/.ssh/id\_rsa -N ''

[root@web2 ~]# ssh-copy-id 192.168.2.100

[root@web2 ~]# git clone root@192.168.2.100:/var/git/base\_ssh

[root@web2 ~]# git push

步骤二:Git协议服务器(只读操作)

[1]安装git-daemon软件包(web1主机)

]#yum -y install git-daemon on

[2]创建版本库(web1主机操作)

]#git init -bare /var/git/base\_ssh

Initialized empty Git repository in /var/git/base\_ssh/

[3]修改配置文件,启动git服务(web1)

[root@web1 ~]# vim /usr/lib/systemd/system/git@.service

修改前内容如下：

ExecStart=-/usr/libexec/git-core/git-daemon --base-path=/var/lib/git

--export-all --user-path=public\_git --syslog --inetd –verbose

修改后内容如下：

ExecStart=-/usr/libexec/git-core/git-daemon --base-path=/var/git

--export-all --user-path=public\_git --syslog --inetd –verbose

[root@web1 ~]# systemctl start git.socket

[4]客户端访问方式(web2)

]#git clone git:192.168.2.100/base\_git

步骤三:HTTP协议服务器(只读)

[1]安装gitweb,httpd软件包(web1)

]#yum -y install httpd gitweb

[2]修改配置文件,设置仓库根目录(web1)

]#vim +11 /etc/gitweb.conf

$projectroot=”/var/git”;

[3]创建版本仓库(web1)

]#git init -bare /var/git/base\_http

[4]启动httpd服务

]#systemctl restart httpd

案例5:制作nginx的RPM

5.1问题

本案例使用nginx-1.12.2版本的远吗软件,生成对应的RPM包软件,具体要求如下:

\*软件名称nginx

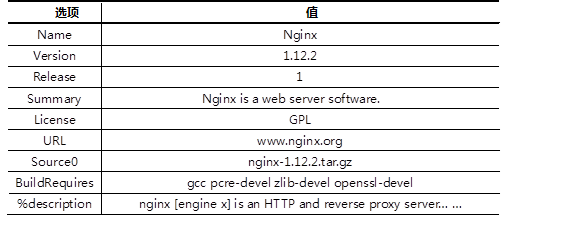
\*软件版本1.12.2

\*RPM软件包可以查询描述信息

\*RPM软件包可以安装及卸载

5.2问题

安装RPM-build软件包,编写SPEC配置文件,创建新的RPM软件包

配置文件中描述信息

5.3步骤

步骤一:安装rpm-buuild软件

[1]安装rpm-buuild软件

]#yum -y install rpm-build

[2]生成rpmbuild目录结构

]#rpmbuild -ba nginx.spec //会报错,没有文件或目录

]#ls /root/rpmbuild //自动生成目录结构

[3]准备工作,将源码文件复制到SOURCES

]#cp ntinx-1.12.2.tsr.gz /root/rpmbuild/SOURES

[4]创建并修改SPEC配置文件

[root@web1 ~]# vim /root/rpmbuild/SPECS/nginx.spec

Name:nginx                                        #源码包软件名称

Version:1.12.2                                    #源码包软件的版本号

Release:    10                                        #制作的RPM包版本号

Summary: Nginx is a web server software.            #RPM软件的概述

License:GPL           #软件的协议

URL:    www.test.com       #网址

Source0:nginx-1.12.2.tar.gz     #源码包文件的全称

#BuildRequires:           #制作RPM时的依赖关系

#Requires:             #安装RPM时的依赖关系

%description

nginx [engine x] is an HTTP and reverse proxy server.    #软件的详细描述

%post

useradd nginx          #非必需操作：安装后脚本(创建账户)

%prep

%setup -q          #自动解压源码包，并cd进入目录

%build

./configure

make %{?\_smp\_mflags}

%install

make install DESTDIR=%{buildroot}

%files

%doc

/usr/local/nginx/\*                    #对哪些文件与目录打包

%changelog

步骤二:使用配置文件床架RPM包

[1]安装依赖软件包

]#yum -y install gcc pcre-devel openssl-devel

[2]rpmbuild创建RPM软件包

[root@web1 ~]# rpmbuild -ba /root/rpmbuild/SPECS/nginx.spec

[root@web1 ~]# ls /root/rpmbuild/RPMS/x86\_64/nginx-1.12.2-10.x86\_64.rpm

步骤三:安装,卸载软件

[root@web1 ~]# rpm -ivh /root/rpmbuild/RPMS/x86\_64/nginx-1.12.2-10.x86\_64.rpm

[root@web1 ~]# rpm -qa |grep nginx

[root@web1 ~]# /usr/local/nginx/sbin/nginx

[root@web1 ~]# curl http://127.0.0.1/