版本控制软件

集中式 : cvs svn

分布式 : git bitkeeper (收费)

集中式 : 开发者之间公用的一个仓库 (repository) 所有操作需要联网

分布式 : 每个开发者都是一个仓库的完整克隆,每个人都是服务器 支持断网操作

git基本概念

git仓库 : 保存所有数据的地方

工作区 : 从仓库中提取出来的文件,放在磁盘上供你使用或修改

暂存区 : 就是一个文件,索引文件,保存了下次将提交的文件列表信息

~]# mkdir /var/git

~]# yum -y install git

~]# git init /var/git/project --bare

初始化空的 Git 版本库于 /var/git/project/

git]# ls /var/git/project/

branches config description HEAD hooks info objects refs

web2 ~]# yum -y install git

本地访问 : git clone file:///var/git

远程ssh访问 : git clone root@服务器ip:/var/git

git支持的子命令

-clone 将远程服务器的仓库克隆到本地

-config 修改git配置

-add 添加修改到暂存区

-commit 提交修改到本地仓库

-push 提交修改到远程服务器

-pull 从远程下载使

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

正克隆到 'project'...

Warning: Permanently added '192.168.2.100' (ECDSA) to the list of known hosts.

root@192.168.2.100's password:

warning: 您似乎克隆了一个空版本库。

config修改git配置(客户端用户标记信息)

git config --global user.email "you@example.com"

git config --global user.name "your name"

.

web2 ~]# ls

project #工作区

web2 ~]# cd project/

web2 project]# ls -a

. .. .git #本地仓库

project]# echo "123456" > index.txt

将暂存区修改提交到本地仓库 : git commit -m "注释,可以为任意字符"

project]# git commit -m "one"

[master（根提交） bdae085] one

2 files changed, 3 insertions(+)

create mode 100644 demo/hosts

create mode 100644 init.txt

\* log日志查看

格式 : git log

例 : project]# git log

commit 57ba4206eaf847e7711ecc533aa9861759bf0e05 #文件的版本号

Author: Your Name <you@example.com>

Date: Wed Nov 13 17:48:20 2019 +0800

two

commit bdae085746174d0a07c2ab6076ebb99c6d7d716f

Author: Your Name <you@example.com>

Date: Wed Nov 13 17:46:17 2019 +0800

one

格式 : git log --pretty=oneline

例 : project]# git log --pretty=oneline

57ba4206eaf847e7711ecc533aa9861759bf0e05 two

bdae085746174d0a07c2ab6076ebb99c6d7d716f one

格式 : git reflog

project]# git reflog

57ba420 HEAD@{0}: commit: two

bdae085 HEAD@{1}: commit (initial): one

project]# git reflog

38dac29 HEAD@{0}: commit: f

37f23f7 HEAD@{1}: commit: e

9c3fd62 HEAD@{2}: commit: d

7d3c8c7 HEAD@{3}: commit: c

801c47d HEAD@{4}: commit: b

2a06a9f HEAD@{5}: commit: a

\* head指针

是一个可以在任何分支和版本移动的指针 通过移动指针可以将数据还原至任何版本

HEAD@{0}指的是当前版本

还原 : git reset -hard 文件的版本号

project]# git reset --hard 9c3fd62

HEAD 现在位于 9c3fd62 d

project]# git reflog

9c3fd62 HEAD@{0}: reset: moving to 9c3fd62

38dac29 HEAD@{1}: commit: f

37f23f7 HEAD@{2}: commit: e

9c3fd62 HEAD@{3}: commit: d

7d3c8c7 HEAD@{4}: commit: c

801c47d HEAD@{5}: commit: b

2a06a9f HEAD@{6}: commit: a

使用head^将版本回滚一个版本

project]# git reset --hard HEAD^

使用head~数字,可以将版本回归n个版本

project]# git reset --hard HEAD~1

\*分支

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

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

git branch -v #列出所有分支

project]# git branch -v

\* master 7d3c8c7 [落后 3] c

git branch 分支名称 #创建分支

project]# git branch hotfix

project]# git branch feature

project]# git checkout hotfix #切换分支

project]# echo "xxx" >> new.txt

[root@web2 project]# git add .

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

[hotfix 7cd54c0] l

1 file changed, 1 insertion(+)

project]# git reflog

7cd54c0 HEAD@{0}: commit: l

38dac29 HEAD@{1}: commit: f

37f23f7 HEAD@{2}: commit: e

9c3fd62 HEAD@{3}: commit: d

7d3c8c7 HEAD@{4}: commit: c

801c47d HEAD@{5}: commit: b

2a06a9f HEAD@{6}: commit: a

project]# git checkout master

切换到分支 'master'

您的分支落后 'origin/master' 共 3 个提交，并且可以快进。

（使用 "git pull" 来更新您的本地分支）

project]# git reflog

7cd54c0 HEAD@{0}: commit: l

38dac29 HEAD@{1}: commit: f

37f23f7 HEAD@{2}: commit: e

9c3fd62 HEAD@{3}: commit: d

7d3c8c7 HEAD@{4}: commit: c

801c47d HEAD@{5}: commit: b

2a06a9f HEAD@{6}: commit: a

project]# git merge hotfix #合并分支 合并前一定要切换到master分支

更新 7d3c8c7..7cd54c0

Fast-forward

new.txt | 1 +

now.txt | 3 +++

2 files changed, 4 insertions(+)

create mode 100644 now.txt

project]# cat new.txt

123

456

second

xxx

\*在不同分支中修改相同文件的相同行数据，模拟数据冲突

project]# git checkout hotfix

切换到分支 'hotfix'

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

[root@web2 project]# git add .

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

[hotfix 222591a] aaa

1 file changed, 1 insertion(+)

create mode 100644 a.txt

[root@web2 project]# git checkout master

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

[root@web2 project]# git add .

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

[master 13bcafd] aaa

1 file changed, 1 insertion(+)

create mode 100644 a.txt

[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。

[root@web2 project]# vim a.txt #修改文件留下要用的

[root@web2 project]# git merge hotfix

error: 'merge' is not possible because you have unmerged files.

提示：请先在工作区改正文件，然后酌情使用

提示：'git add/rm <file>' 标记解决方案，

提示：或使用 'git commit -a'。

fatal: Exiting because of an unresolved conflict.

[root@web2 project]# cat a.txt

BBB

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

分支操作流程如图-6，图-7，图-8，图-9，图-10所示。

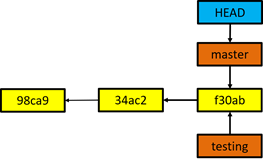


图-6 HEAD指针指向master分支

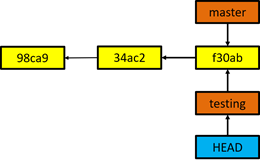


图-7 切换分支，HEAD指针指向testing分支

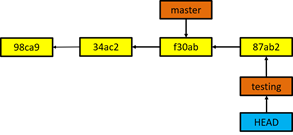


图-8 在testing分支中修改并提交代码

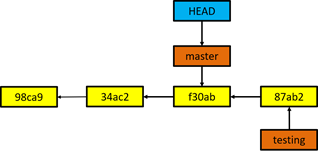


图-9 将分支切换回master分支

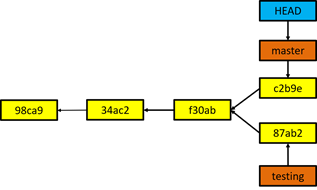


图-10 在master分支中修改数据，更新版本

Git支持很多服务器协议形式，不同协议的Git服务器，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（支持读写操作）**

1）创建基于密码验证的SSH协议服务器（web1主机操作）。

1. **[**root@web1 **~]**# git init **--**bare **/var**/git/base\_ssh
2. Initialized empty Git repository **in** /var/git**/**base\_ssh**/**

2)客户端访问的方式（web2主机操作）。

1. **[**root@web2 **~]**# git clone root@**192.168.2.100:**/var/git**/**base\_ssh
2. **[**root@web2 **~]**# rm **-**rf base\_ssh

3）客户端生成SSH密钥，实现免密码登陆git服务器（web2主机操作）。

1. **[**root@web2 **~]**# ssh**-**keygen **-**f **/**root**/.**ssh**/**id\_rsa **-**N ''
2. **[**root@web2 **~]**# ssh**-**copy**-**id **192.168.2.100**
3. **[**root@web2 **~]**# git clone root@**192.168.2.100:**/var/git**/**base\_ssh
4. **[**root@web2 **~]**# git push

**步骤二：Git协议服务器（只读操作的服务器）**

1）安装git-daemon软件包（web1主机操作）。

1. **[**root@web1 **~]**# yum **-**y install git**-**daemon

2）创建版本库（web1主机操作）。

1. **[**root@web1 **~]**# git init **--**bare **/var**/git/base\_git
2. Initialized empty Git repository **in** /var/git**/**base\_git**/**

3）修改配置文件，启动git服务（web1主机操作）。

1. **[**root@web1 **~]**# vim **/**usr**/**lib**/**systemd**/**system**/**git@**.**service
2. 修改前内容如下：
3. ExecStart**=-**/usr/libexec**/**git**-**core**/**git**-**daemon **--**base**-**path**=**/var/lib**/**git
4. **--export-**all **--**user**-**path**=**public\_git **--**syslog **--**inetd –verbose
5. 修改后内容如下：
6. ExecStart**=-**/usr/libexec**/**git**-**core**/**git**-**daemon **--**base**-**path**=**/var/git
7. **--export-**all **--**user**-**path**=**public\_git **--**syslog **--**inetd –verbose
8. **[**root@web1 **~]**# systemctl start git**.**socket

4）客户端访问方式（web2主机操作）

1. **[**root@web2 **~]**# git clone git**:**//192.168.2.100/base\_git

**HTTP协议服务器（只读操作的服务器）**

1）安装gitweb、httpd软件包（web1主机操作）。

1. **[**root@web1 **~]**# yum **-**y install httpd gitweb

2）修改配置文件，设置仓库根目录（web1主机操作）。

1. **[**root@web1 **~]**# vim **+11** **/**etc**/**gitweb**.**conf
2. $projectroot **=** "/var/git"**;**                        #添加一行

3) 创建版本仓库（web1主机操作）

1. **[**root@web1 **~]**# git init **--**bare **/var**/git/base\_http

4）启动httpd服务器

1. **[**root@web1 **~]**# systemctl start httpd

5）客户端访问方式（web2主机操作）

注意：调用虚拟机中的firefox浏览器，需要在远程时使用ssh -X 服务器IP，并且确保真实主机的firefox已经关闭。

1. **[**root@web2 **~]**# firefox http**:**//192.168.2.100/git/

**课外扩展知识：注册使用Github**

1.登陆网站https://github.com，点击Sign up（注册），如图-11所示。

IMG_256

图-11

2.填写注册信息（用户名，邮箱，密码），如图-12所示。

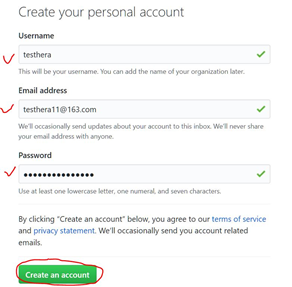


图-12

3. 初始化操作，如图-13和图-14所示。

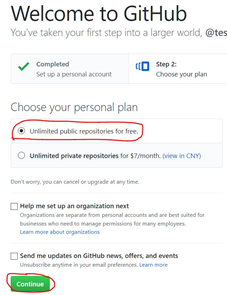


图-13

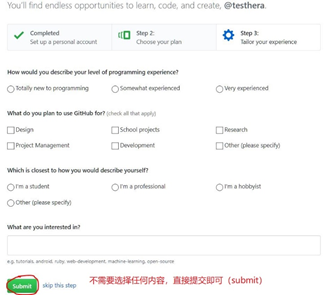


图-14

注意，初始化完成后，到邮箱中去激活Github账户。

4. 创建仓库、使用仓库

点击Start a project（如图-15所示），

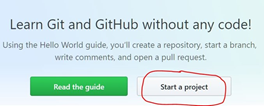


图-15

填写项目名称（项目名称任意），如图-16所示。

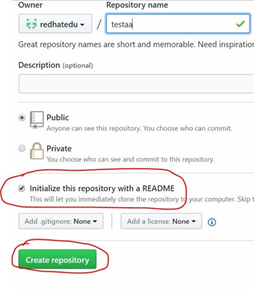


图-16

往仓库中上传文件或新建文件，如图-17所示

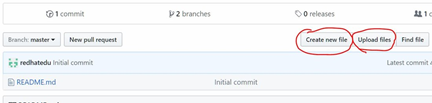


图-17

下载仓库中的代码，如图-18所示。

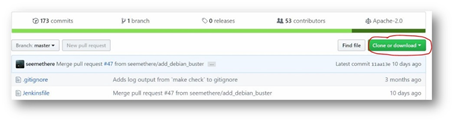


图-18

5. 命令行操作（需要联网的主机，如真实机）

[root@pc001 ~]# yum -y install git

[root@pc001 ~]# git clone https://github.com/账户名称/仓库名称

#clone指令用于将服务器仓库中的资料打包下载到本地

[root@pc001 ~]# cd 仓库名称

[root@pc001 ~]# 任意修改文件，或新建文件

[root@pc001 ~]# git add .

#add添加新文件

[root@pc001 ~]# git commit -m "test"

[root@pc001 ~]# git push

#commit和push实现提交代码的功能

[root@pc001 ~]# git pull

#pull可以从githuab服务器拉取数据到本地

## **5 案例5：制作nginx的RPM包**

### **5.1 问题**

本案例使用nginx-1.12.2版本的源码软件，生成对应的RPM包软件，具体要求如下：

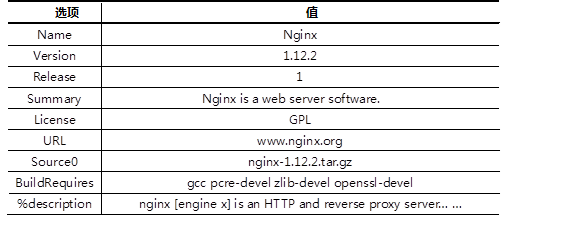
* 软件名称为nginx
* 软件版本为1.12.2
* RPM软件包可以查询描述信息
* RPM软件包可以安装及卸载

### **5.2 方案**

安装rpm-build软件包，编写SPEC配置文件，创建新的RPM软件包。

配置文件中的描述信息如表-2：

表－2 SPEC描述信息



### **5.3 步骤**

实现此案例需要按照如下步骤进行。

**步骤一：安装rpm-build软件**

1）安装rpm-build软件包

1. **[**root@web1 **~]**# yum **-**y install rpm**-**build

2）生成rpmbuild目录结构

1. **[**root@web1 **~]**# rpmbuild **-**ba nginx**.**spec                //会报错，没有文件或目录
2. **[**root@web1 **~]**# ls **/**root**/**rpmbuild                    //自动生成的目录结构
3. BUILD BUILDROOT RPMS SOURCES SPECS SRPMS

3）准备工作，将源码软件复制到SOURCES目录

1. **[**root@web1 **~]**# cp nginx**-1.12.2.**tar**.**gz **/**root**/**rpmbuild**/**SOURCES**/**

4）创建并修改SPEC配置文件

1. **[**root@web1 **~]**# vim **/**root**/**rpmbuild**/**SPECS**/**nginx**.**spec
2. Name**:**nginx                                        #源码包软件名称
3. Version**:1.12.2**                                    #源码包软件的版本号
4. Release**:**    **10**                                        #制作的RPM包版本号
5. Summary**:** Nginx is a web server software**.**            #RPM软件的概述
6. License**:**GPL                                        #软件的协议
7. URL**:**    www**.**test**.**com                                    #网址
8. Source0**:**nginx**-1.12.2.**tar**.**gz                        #源码包文件的全称
9. #BuildRequires**:**                                    #制作RPM时的依赖关系
10. #Requires**:**                                        #安装RPM时的依赖关系
11. **%**description
12. nginx **[**engine x**]** is an HTTP and reverse proxy server**.**    #软件的详细描述
13. **%**post
14. useradd nginx          #非必需操作：安装后脚本**(**创建账户**)**
15. **%**prep
16. **%**setup **-**q                                #自动解压源码包，并cd进入目录
17. **%**build
18. **./**configure
19. make **%{?**\_smp\_mflags**}**
20. **%**install
21. make install DESTDIR**=%{**buildroot**}**
22. **%**files
23. **%**doc
24. /usr/local**/**nginx/\*                    #对哪些文件与目录打包
25. %changelog

**步骤二：使用配置文件创建RPM包**

1）安装依赖软件包

1. **[**root@web1 **~]**# yum **-**y install gcc pcre**-**devel openssl**-**devel

2）rpmbuild创建RPM软件包

1. **[**root@web1 **~]**# rpmbuild **-**ba **/**root**/**rpmbuild**/**SPECS**/**nginx**.**spec
2. **[**root@web1 **~]**# ls **/**root**/**rpmbuild**/**RPMS**/**x86\_64**/**nginx**-1.12.2-10.**x86\_64**.**rpm

**步骤三：安装软件**

1. **[**root@web1 **~]**# yum install **/**root**/**rpmbuild**/**RPMS**/**x86\_64**/**nginx**-1.12.2-10.**x86\_64**.**rpm
2. **[**root@web1 **~]**# rpm **-**qa **|**grep nginx
3. **[**root@web1 **~]**# ls **/**usr**/**local**/**nginx**/**