[Top](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/OPERATION/DAY06/CASE/01/index.html" \l "page_top_case)

# NSD OPERATION DAY06

1. [案例1：Git基本操作](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/OPERATION/DAY06/CASE/01/index.html" \l "case1)
2. [案例2： HEAD指针操作](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/OPERATION/DAY06/CASE/01/index.html" \l "case2)
3. [案例3：Git分支操作](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/OPERATION/DAY06/CASE/01/index.html" \l "case3)
4. [案例4：Git服务器](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/OPERATION/DAY06/CASE/01/index.html" \l "case4)
5. [案例5：制作nginx的RPM包](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/OPERATION/DAY06/CASE/01/index.html" \l "case5)

## 1 案例1：Git基本操作

### 1.1 问题

本案例要求先快速搭建好一台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所示，Git工作流如图-2所示。

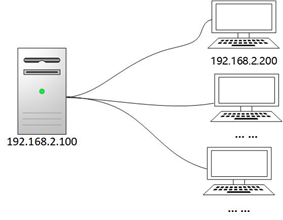


图-1

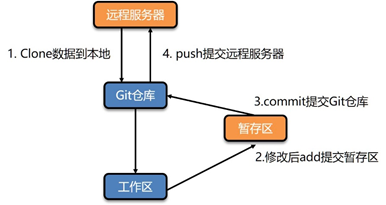


图-2

### 1.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：部署Git服务器（192.168.2.100作为远程git服务器）

1）YUM安装Git软件。

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

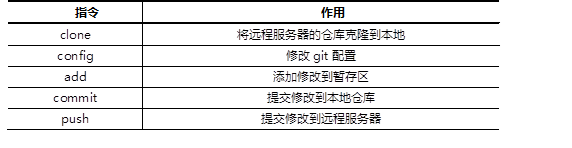
2)初始化一个空仓库。

1. [root@web1 ~]# mkdir /var/git
2. [root@web1 ~]# git init /var/git/project --bare
3. [root@web1 ~]# ls /var/git/project
4. 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常用指令列表如表-1所示。

表－1 git常用指令列表



1) clone克隆服务器仓库到本地。

1. [root@web2 ~]# yum -y install git
2. [root@web2 ~]# git clone root@192.168.2.100:/var/git/project
3. [root@web2 ~]# cd project
4. [root@web2 ~]# ls

2) 修改git配置。

1. [root@web2 project]# git config --global user.email "you@example.com"
2. [root@web2 project]# git config --global user.name "Your Name"
3. [root@web2 project]# cat ~/.gitconfig
4. [user]
5. email = you@example.com
6. name = Your Name

3） 本地工作区对数据进行增删改查(必须要先进入仓库再操作数据)。

1. [root@web2 project]# echo "init date" > init.txt
2. [root@web2 project]# mkdir demo
3. [root@web2 project]# cp /etc/hosts demo

4） 查看仓库中数据的状态。

1. [root@web2 project]# git status

5） 将工作区的修改提交到暂存区。

1. [root@web2 project]# git add .

6) 将暂存区修改提交到本地仓库。

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

7） 将本地仓库中的数据推送到远程服务器(web2将数据推送到web1)。

1. [root@web2 project]# git config --global push.default simple
2. [root@web2 project]# git push
3. root@192.168.2.100's password: 输入服务器root密码
4. [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。

1. [root@web2 project]# git pull

9) 查看版本日志。

1. [root@web2 project]# git log
2. [root@web2 project]# git log --pretty=oneline
3. [root@web2 project]# git log --oneline
4. [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。如图-3、图-4所示。



图-3



图-4

## 2 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指针是一个可以在任何分支和版本移动的指针，通过移动指针我们可以将数据还原至任何版本。每做一次提交操作都会导致git更新一个版本，HEAD指针也跟着自动移动。

### 2.3 步骤

实现此案例需要按照如下步骤进行。

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）。

1. [root@web2 project]# echo "new file" > new.txt
2. [root@web2 project]# git add .
3. [root@web2 project]# git commit -m "add new.txt"
4. [root@web2 project]# echo "first" >> new.txt
5. [root@web2 project]# git add .
6. [root@web2 project]# git commit -m "new.txt:first line"
7. [root@web2 project]# echo "second" >> new.txt
8. [root@web2 project]# git add .
9. [root@web2 project]# git commit -m "new.txt:second"
10. [root@web2 project]# echo "third" >> new.txt
11. [root@web2 project]# git add .
12. [root@web2 project]# git commit -m "new.txt:third"
13. [root@web2 project]# git push
14. [root@web2 project]# echo "123" > num.txt
15. [root@web2 project]# git add .
16. [root@web2 project]# git commit -m "num.txt:123"
17. [root@web2 project]# echo "456" > num.txt
18. [root@web2 project]# git add .
19. [root@web2 project]# git commit -m "num.txt:456"
20. [root@web2 project]# echo "789" > num.txt
21. [root@web2 project]# git add .
22. [root@web2 project]# git commit -m "num.txt:789"
23. [root@web2 project]# git push

2) 查看Git版本信息。

1. [root@web2 project]# git reflog
2. [root@web2 project]# git log --oneline
3. 04ddc0f num.txt:789
4. 7bba57b num.txt:456
5. 301c090 num.txt:123
6. b427164 new.txt:third
7. 0584949 new.txt:second
8. ece2dfd new.txt:first line
9. e1112ac add new.txt
10. 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}。

1. [root@web2 project]# git reset --hard 301c0
2. [root@web2 project]# git reflog
3. 301c090 HEAD@{0}: reset: moving to 301c0
4. 04ddc0f HEAD@{1}: commit: num.txt:789
5. 7bba57b HEAD@{2}: commit: num.txt:456
6. 301c090 HEAD@{3}: commit: num.txt:123
7. b427164 HEAD@{5}: commit: new.txt:third
8. 0584949 HEAD@{6}: commit: new.txt:second
9. ece2dfd HEAD@{7}: commit: new.txt:first line
10. e1112ac HEAD@{8}: commit: add new.txt
11. 1a0d908 HEAD@{9}: commit (initial): 初始化
12. [root@web2 project]# cat num.txt                #查看文件是否为123
13. 123
14. [root@web2 project]# git reset --hard 7bba57b
15. [root@web2 project]# cat num.txt                #查看文件是否为123，456
16. 123
17. 456
18. [root@web2 project]# git reflog                #查看指针移动历史
19. 7bba57b HEAD@{0}: reset: moving to 7bba57b
20. 301c090 HEAD@{1}: reset: moving to 301c0
21. … …
22. [root@web2 project]# git reset --hard 04ddc0f    #回到最后一次修改的版本

## 3 案例3：Git分支操作

### 3.1 问题

沿用练习二，学习操作Git分支，具体要求如下：

* 查看分支
* 创建分支
* 切换分支
* 合并分支
* 解决分支的冲突

### 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-5所示。

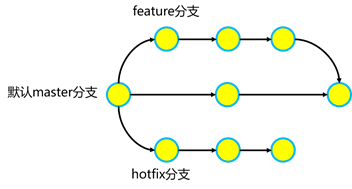


图-5

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：查看并创建分支

1）查看当前分支。

1. [root@web2 project]# git status
2. # On branch master
3. nothing to commit, working directory clean
4. [root@web2 project]# git branch -v
5. \* master 0dc2b76 delete init.txt

2）创建分支。

1. [root@web2 project]# git branch hotfix
2. [root@web2 project]# git branch feature
3. [root@web2 project]# git branch -v
4. feature 0dc2b76 delete init.txt
5. hotfix 0dc2b76 delete init.txt
6. \* master 0dc2b76 delete init.txt

步骤二：切换与合并分支

1）切换分支。

1. [root@web2 project]# git checkout hotfix
2. [root@web2 project]# git branch -v
3. feature 0dc2b76 delete init.txt
4. \* hotfix 0dc2b76 delete init.txt
5. 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、查）。

1. [root@web2 project]# echo "fix a bug" >> new.txt
2. [root@web2 project]# git add .
3. [root@web2 project]# git commit -m "fix a bug"

3）将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。

1. [root@web2 project]# git checkout master
2. [root@web2 project]# cat new.txt        #默认master分支中没有hotfix分支中的数据
3. [root@web2 project]# git merge hotfix
4. Updating 0dc2b76..5b4a755
5. Fast-forward
6. new.txt | 1 ++
7. 1 file changed, 1 insertions(+)

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。

1. [root@web2 project]# git checkout hotfix
2. [root@web2 project]# echo "AAA" > a.txt
3. [root@web2 project]# git add .
4. [root@web2 project]# git commit -m "add a.txt by hotfix"
5. [root@web2 project]# git checkout master
6. [root@web2 project]# echo "BBB" > a.txt
7. [root@web2 project]# git add .
8. [root@web2 project]# git commit -m "add a.txt by master"
9. [root@web2 project]# git merge hotfix
10. 自动合并 a.txt
11. 冲突（添加/添加）：合并冲突于 a.txt
12. 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。

1. [root@web2 project]# cat a.txt                #该文件中包含有冲突的内容
2. <<<<<<< HEAD
3. BBB
4. =======
5. AAA
6. >>>>>>> hotfix
7. [root@web2 project]# vim a.txt #修改该文件，为最终需要的数据，解决冲突
8. BBB
9. [root@web2 project]# git add .
10. [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

分支操作流程如图-6，图-7，图-8，图-9，图-10所示。

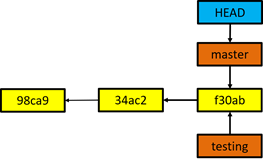


图-6 HEAD指针指向master分支

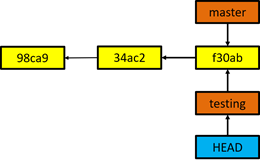


图-7 切换分支，HEAD指针指向testing分支

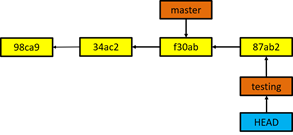


图-8 在testing分支中修改并提交代码

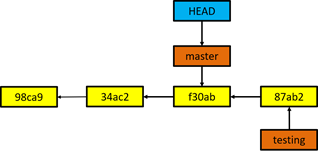


图-9 将分支切换回master分支

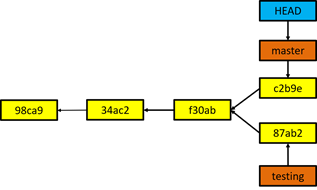


图-10 在master分支中修改数据，更新版本

## 4 案例4：Git服务器

### 4.1 问题

沿用练习三，学习Git不同的服务器形式，具体要求如下：

* 创建SSH协议服务器
* 创建Git协议服务器
* 创建HTTP协议服务器

### 4.2 方案

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

步骤三：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/

步骤四：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_267

图-11

2.填写注册信息（用户名，邮箱，密码），如图-12所示。

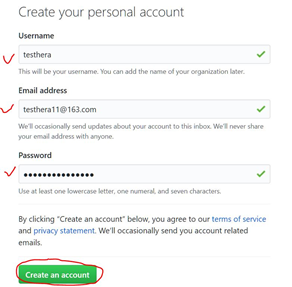


图-12

3. 初始化操作，如图-13和图-14所示。

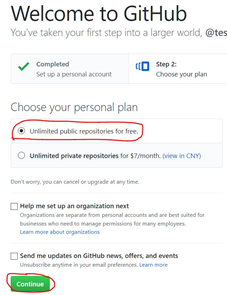


图-13

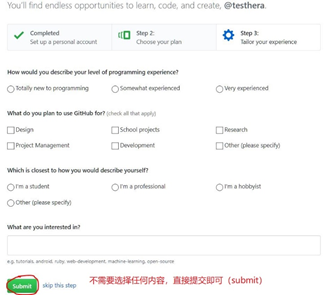


图-14

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），

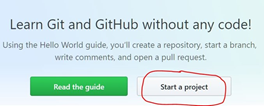


图-15

填写项目名称（项目名称任意），如图-16所示。

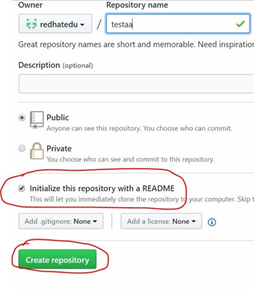


图-16

往仓库中上传文件或新建文件，如图-17所示

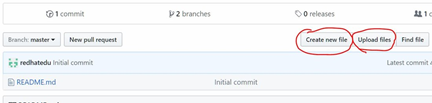


图-17

下载仓库中的代码，如图-18所示。

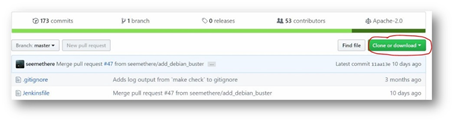


图-18

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：

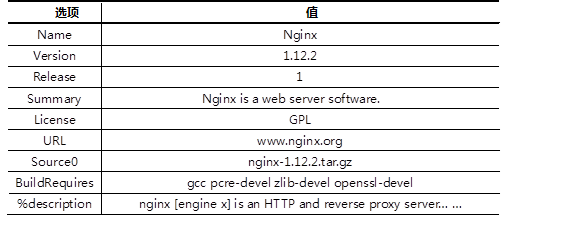
* 软件名称为nginx
* 软件版本为1.12.2
* RPM软件包可以查询描述信息
* RPM软件包可以安装及卸载

### 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/