

Linux高级运维

NSD OPERATION

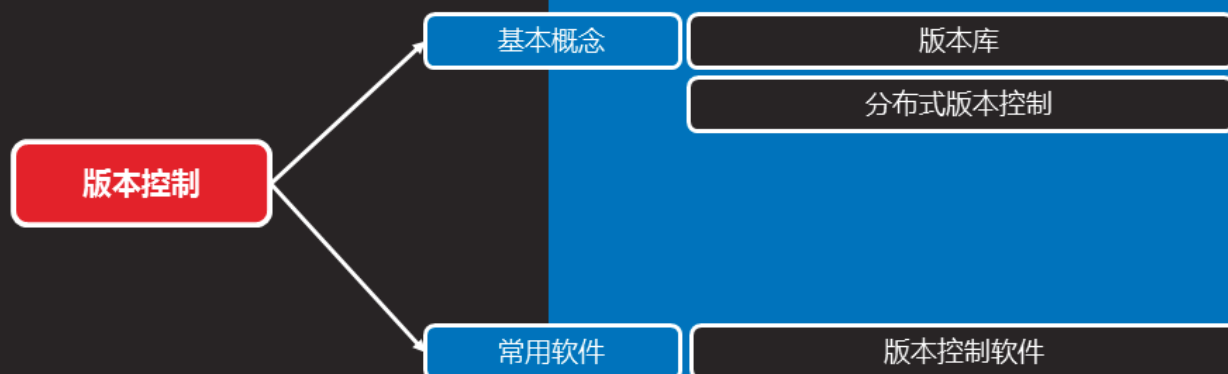
DAY06

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	版本控制
	10:30 ~ 11:20	Git基础
	11:30 ~ 12:20	
下午	14:00 ~ 14:50	Git进阶
	15:00 ~ 15:50	
	16:00 ~ 16:50	RPM打包
	17:00 ~ 17:30	总结和答疑



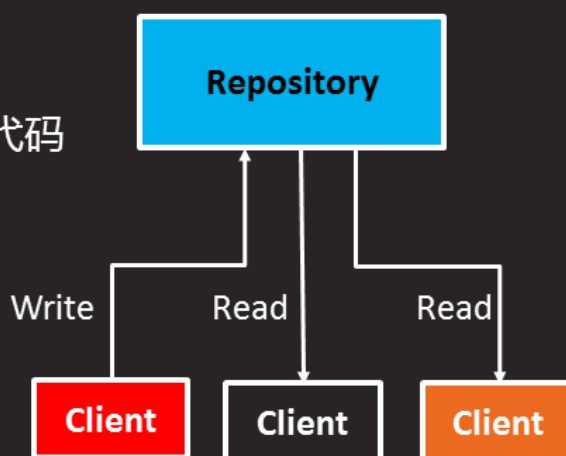
版本控制



基本概念

版本库

- 典型的客户/服务器系统
 - 版本库是版本控制的核心
 - 任意数量客户端
 - 客户端通过写数据库分享代码



分布式版本控制

知识讲解

- 集中式版本控制系统
 - 开发者之间共用一个仓库 (repository)
 - 所有操作需要联网
- 分布式版本控制系统
 - 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



分布式版本控制（续1）

知识讲解

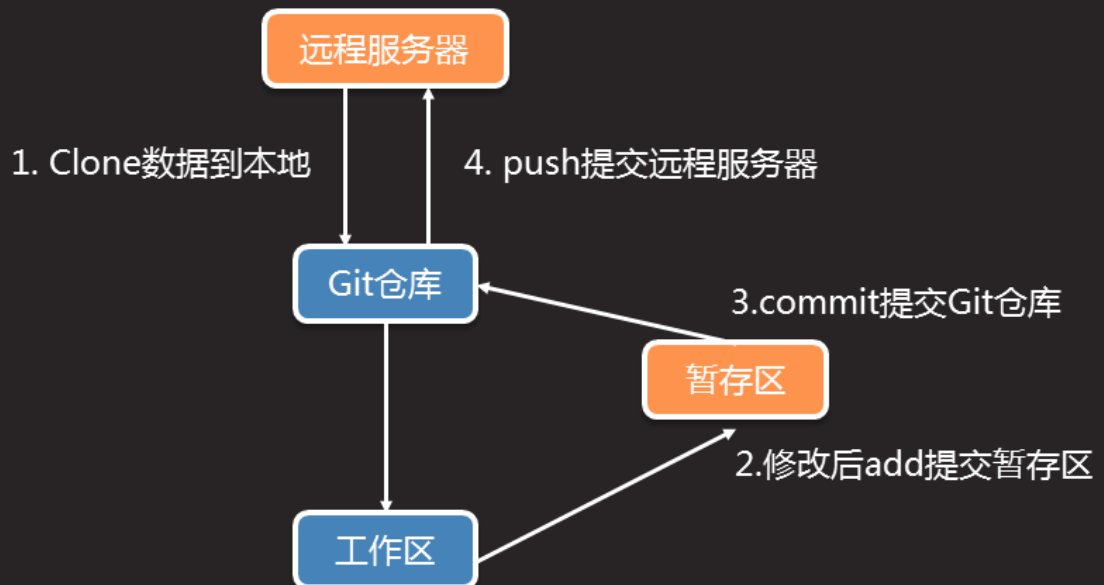
- Git基本概念
 - Git仓库：
保存所有数据的地方
 - 工作区：
从仓库中提取出来的文件，放在磁盘上供你使用或修改
 - 暂存区：
就是一个文件，索引文件，保存了下次将提交的文件列表信息



分布式版本控制（续2）

- 工作流

知识讲解



常用软件

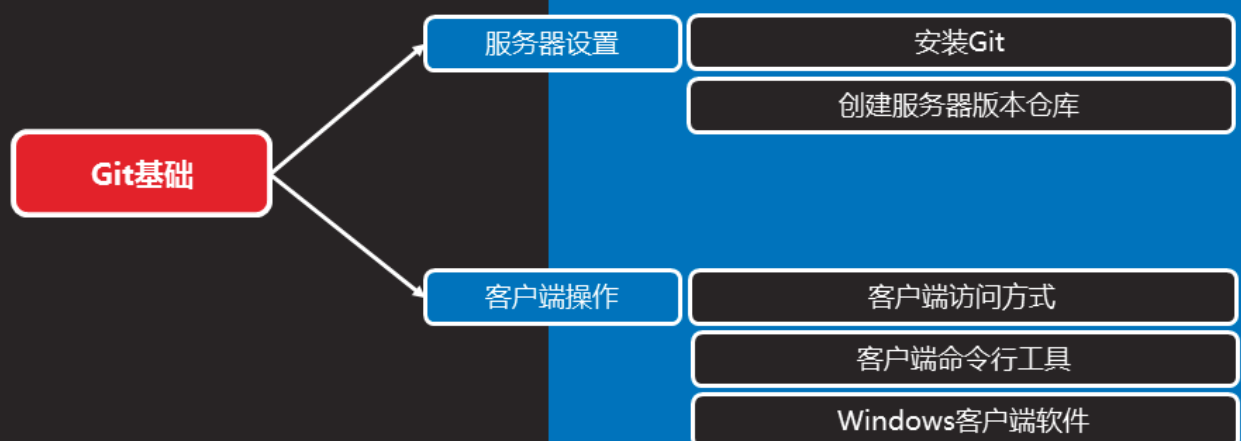
版本控制软件

知识讲解

- 集中式版本控制软件
 - CVS
 - SVN (Subversion)
- 分布式版本控制软件
 - Git
 - BitKeeper (收费)



Git基础



服务器设置

安装Git

- 安装Git软件

```
[root@svr5 ~]# yum -y install git
```

```
[root@svr5 ~]# git --version  
git version 1.8.3.1
```

创建服务器版本仓库

- 服务器是一台多人协作的中心服务器
 - init初始化一个空仓库（没有具体数据）

知识讲解

```
[root@svr5 ~]# git init /var/git --bare
[root@svr5 ~]# ls /var/git/
config  description  HEAD  hooks  info  objects  refs
```



客户端操作

客户端访问方式

知识讲解

- 本地访问
 - `git clone file:///var/git`
- 远程ssh访问
 - `git clone root@服务器IP:/var/git`
- Web
 - 服务器需要额外配置Web服务器
 - 客户端可以浏览器访问
 - `git clone http://服务器IP/git仓库`
 - `Git clone https://服务器IP/git仓库`



客户端命令行工具

知识讲解

- git支持的子命令操作：
 - 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



客户端命令行工具（续1）

- Clone克隆服务器仓库到本地

知识讲解

```
[root@client1 ~]# yum -y install git
[root@client1 ~]# git clone root@服务器IP:/var/git
Cloning into 'git'...
root@服务器IP's password:
warning: You appear to have cloned an empty repository.
```

```
[root@client1 ~]# ls
anaconda-ks.cfg  git
```



客户端命令行工具（续2）

- config修改git配置
 - 客户端用户标记信息（跟Git账户和密码无关）

知识讲解

```
[root@client1 ~]# git config --global user.email "you@example.com"
[root@client1 ~]# git config --global user.name "Your Name"
[root@client1 ~]# cat .gitconfig
[user]
    email = you@example.com
    name = Your Name
```



客户端命令行工具（续3）

- 导入数据，add提交本地暂存区
 - 提示：必须进入工作目录git操作

知识讲解

```
[root@client1 ~]# cd git
[root@client1 git]# echo "hello" > test.txt
[root@client1 git]# mkdir demo
[root@client1 git]# cp /etc/hosts demo/
[root@client1 git]# git status
[root@client1 git]# git add .
```



客户端命令行工具（续4）

- commit提交本地仓库

知识讲解

```
[root@client1 git]# git commit -m "注释"
[master (根提交) fad80f8] 注释
2 files changed, 3 insertions(+)
create mode 100644 demo/hosts
create mode 100644 test.txt
[root@client1 git]# git status
```



客户端命令行工具（续5）

知识讲解

- push将本地修改提交远程服务器仓库
 - push.default定义如何推送（更安全地推送）
[root@client1 git]# git config --global push.default simple
[root@client1 git]# git push
root@服务器ip's password: 输入服务器root密码
Counting objects: 5, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 357 bytes | 0 bytes/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To root@服务器ip:/var/git
* [new branch] master -> master
[root@client1 git]# git status
- 从远程下载使用pull命令
 - git pull

+

客户端命令行工具（续6）

知识讲解

- log查看历史日志
[root@client1 git]# git log
commit 024f46d4d763ec185fbaeaf73701ea2ee01ea1ef
Merge: fe5a436 008ef54
Author: Your Name <you@example.com>
Date: Tue Feb 19 04:27:41 2019 -0500
[root@client1 git]# git log --pretty=oneline
024f46d4d763ec185fbaeaf73701ea2ee01ea1ef modify test.txt
fe5a4363573c8179d16257c8769c63d86754d827 add a.txt file
008ef548415fb1494e46747759ae4458f0668017 xx
e5c75c9ebe18375822ef5cfab1d0261fb47a2506 1
[root@client1 git]# git reflog
1fac48c HEAD@{0}: commit: modify test.txt
bde26d2 HEAD@{1}: commit: add a.txt file
5ec6b76 HEAD@{2}: commit (initial): test

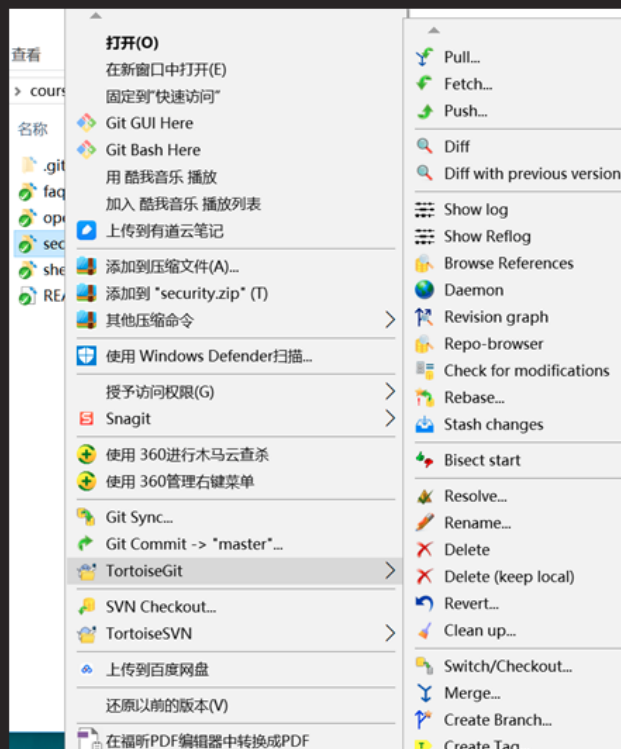
#方便后期回滚数据

+

Windows客户端软件

- 需要安装git和tortoiseGit

知识讲解



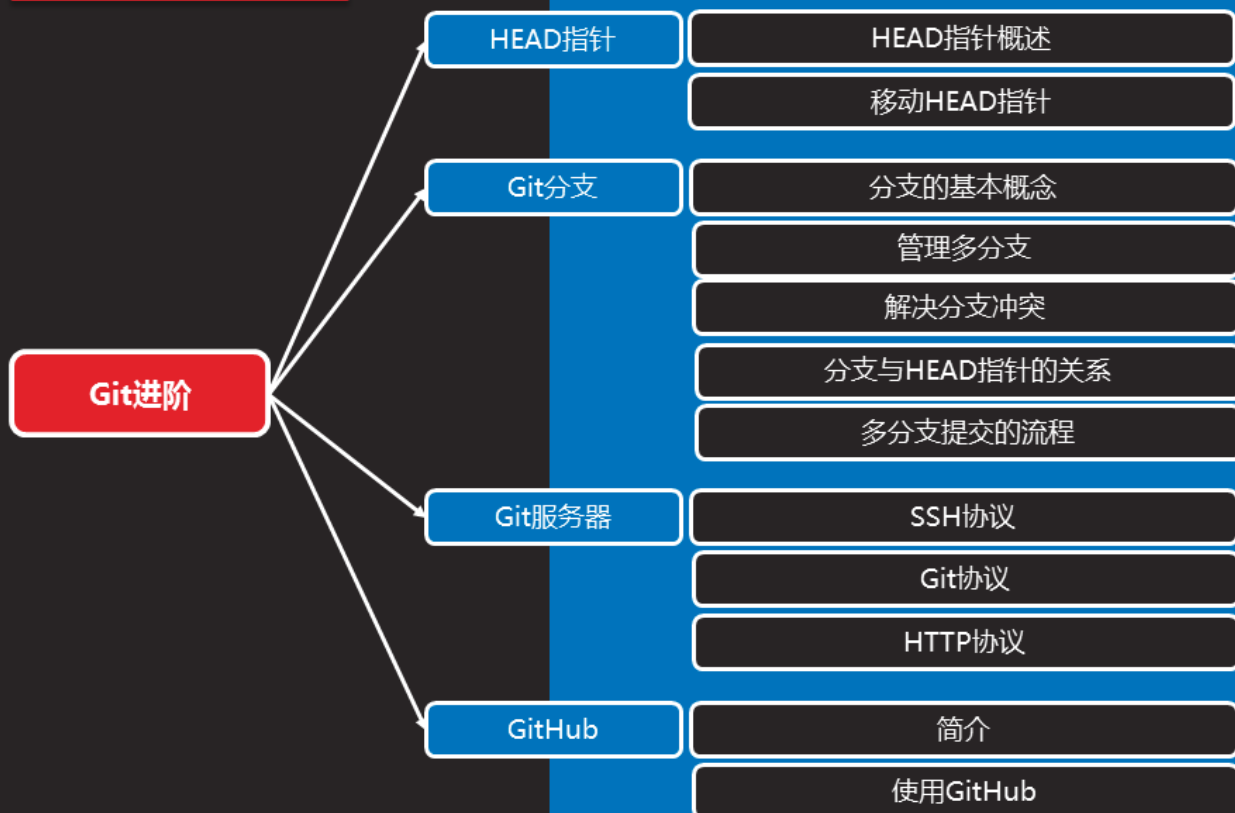
案例1：Git基本操作

- 安装Git软件
 - 创建服务器Git仓库
- 客户端使用git连接服务器进行基本操作
 - 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

课堂练习



Git进阶



HEAD指针

HEAD指针概述

- 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

知识讲解

HEAD指针



```
[root@client1 git]# git log --pretty=oneline
51cbd380cd49c6c187c0fcf2aff8b1ec3ec33936 add new file b.txt
1fac48cd0e9fdf70b13252c63fba92e2a55ef531 modify test.txt
bde26d237962b93d04d3af916a6be6b2ae9bcfca add a.txt file
5ec6b76e5ed3cd180e886e17066e79cd0b827346 test
```



移动HEAD指针

- 先使用log指令查看版本信息

```
[root@client1 git]# git log --pretty=oneline
51cbd380cd49c6c187c0fcf2aff8b1ec3ec33936 add new file b.txt
1fac48cd0e9fdf70b13252c63fba92e2a55ef531 modify test.txt
bde26d237962b93d04d3af916a6be6b2ae9bcfca add a.txt file
5ec6b76e5ed3cd180e886e17066e79cd0b827346 test
```

- 我们需要回到bde26这个版本

```
[root@client1 git]# git reset --hard bde26
HEAD 现在位于 bde26d2 add a.txt file
```

- 在使用ls查看当前工作目录的资料已经还原

```
[root@client1 git]# ls
```

知识讲解



移动HEAD指针（续1）

知识讲解

- 使用HEAD^将版本回滚一个版本

```
[root@client1 git]# git reset --hard HEAD^  
HEAD 现在位于 5ec6b76 test
```

- 使用HEAD~数字，可以将版本回归n个版本

```
[root@client1 git]# git reset --hard HEAD~2  
HEAD 现在位于 bde26d2 add a.txt file
```

- 默认log仅显示当前到之前的版本信息，--all查看所有

```
[root@client1 git]# git log --all --pretty=oneline  
51cbd380cd49c6c187c0fcf2aff8b1ec3ec33936 add new file b.txt  
1fac48cd0e9fdf70b13252c63fba92e2a55ef531 modify test.txt  
bde26d237962b93d04d3af916a6be6b2ae9bcfca add a.txt file
```



案例2：HEAD指针操作

课堂练习

- 查看Git版本信息
- 移动指针
- 通过移动HEAD指针恢复数据
- 合并版本

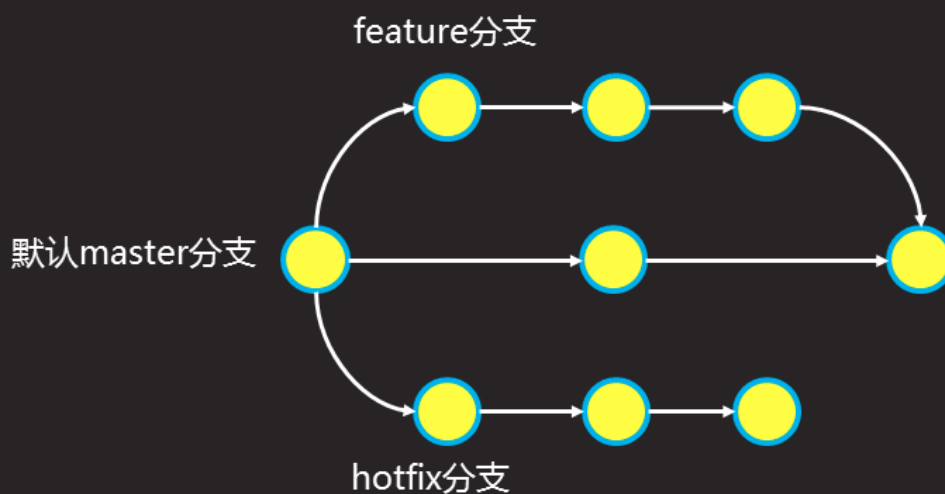


Git分支

分支的基本概念

- 分支可以让开发分多条主线同时进行，每条主线互不影响
 - 按功能模块分支、按版本分支
 - 分支也可以合并

知识讲解



分支的基本概念（续1）

知识讲解

- 常见的分支规范
 - 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）



管理多分支

知识讲解

- 查看当前分支

```
[root@client1 git]# git status
# 位于分支 master
无文件要提交，干净的工作区
[root@client git]# git branch -v
* master 470f049
```
- 创建分支

```
[root@client1 git]# git branch hotfix
[root@client1 git]# git branch feature
[root@client1 git]# git branch -v
feature 470f049
hotfix 470f049
* master 470f049
```



管理多分支（续1）

知识讲解

- 切换分支

```
[root@client1 git]# git checkout hotfix
切换到分支 'hotfix'
[root@client1 git]# git branch -v
  feature 470f049
* hotfix  470f049
  master  470f049
```
- 在新的分支上就可以继续修改代码
 - 修改文件、创建文件等操作
 - 正常add, commit提交版本库



管理多分支（续2）

知识讲解

- 将hotfix合并到master分支
 - 合并前, 一定要切换到master分支
 - 执行merge命令合并分支

```
[root@client1 git]# git checkout master
切换到分支 'master'
[root@client1 git]# git merge hotfix
更新 470f049..79d0ec1
Fast-forward
 a.txt | 1 +
 bug.txt | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 bug.txt
```



解决分支冲突

知识讲解

- 修改了不同分支中相同文件的相同行
 - 系统无法合并分支，产生了冲突

```
[root@client1 git]# git checkout hotfix           #切换至hotfix分支
[root@client1 git]# vim a.txt
[root@client1 git]# git add a.txt                 #修改文件并提取
[root@client1 git]# git commit -m "hotfix"
[root@client1 git]# git checkout master          #切换分支
[root@client1 git]# vim a.txt
[root@client1 git]# git add a.txt
[root@client1 git]# git commit -m "hotfix"
[root@client1 git]# git merge hotfix             #合并时冲突
自动合并 a.txt
冲突（内容）： 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。
```



解决分支冲突（续1）

知识讲解

- 查看有冲突的文件

```
[root@client1 git]# cat a.txt
<<<<<<< HEAD                                     #当前分支的内容
master
=====
hotfix
>>>>>>> hotfix                                   #其他分支的内容
bug
```

- 直接修改有冲突的文件，修改为最终需要的文件内容
 - 修改完成后，正常add，commit提交，解决冲突

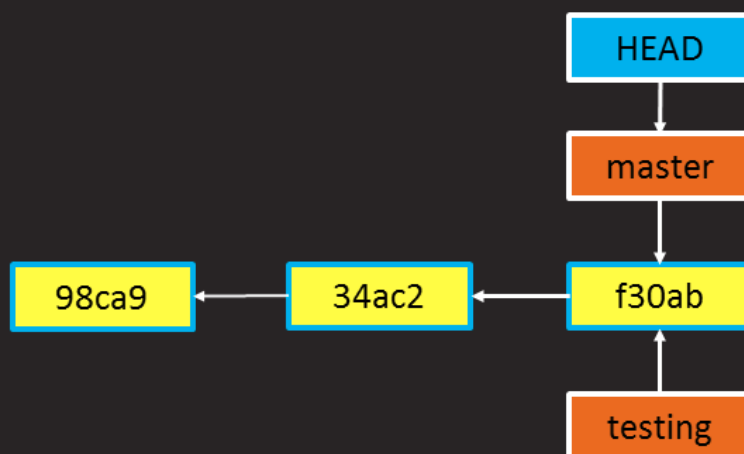
```
[root@client1 git]# git add .
[root@client1 git]# git commit -m "resolve"
```



分支与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？
 - 根据HEAD这个特殊指针

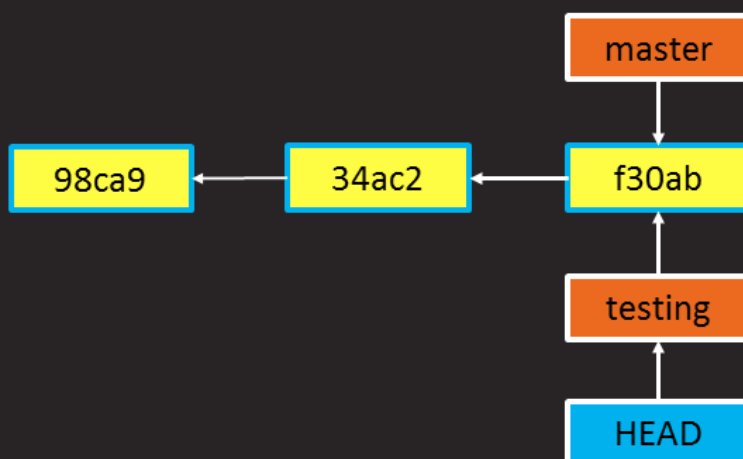
知识讲解



多分支提交的流程

- 切换分支，只是移动HEAD指针

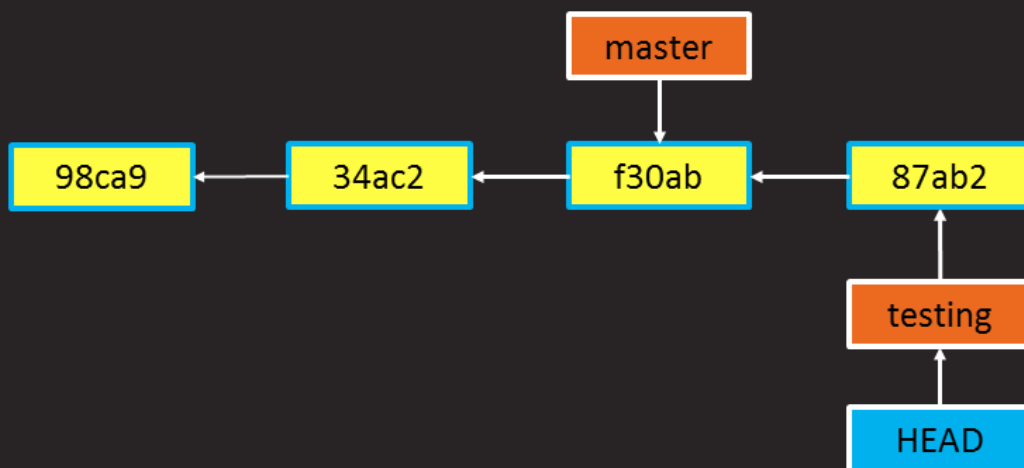
知识讲解



多分支提交的流程（续1）

- testing分支的提交，不会影响master

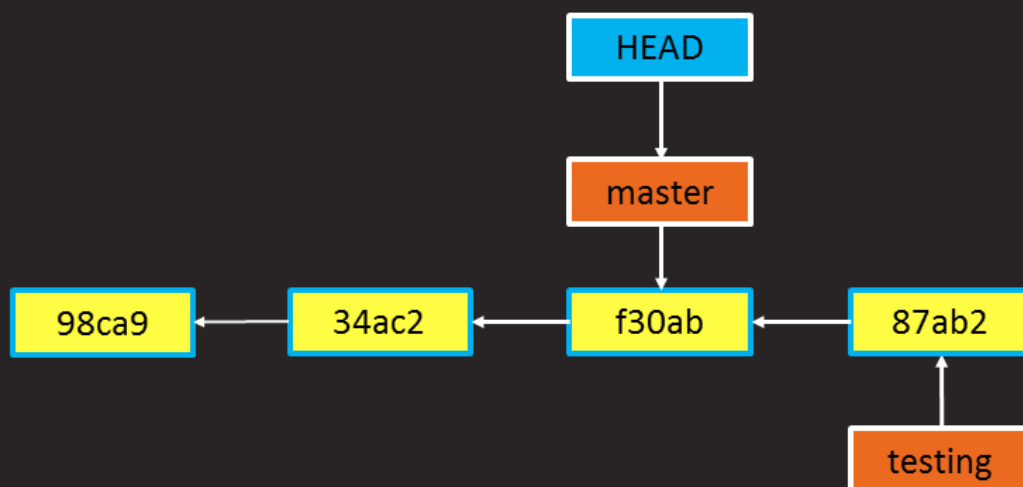
知识讲解



多分支提交的流程（续2）

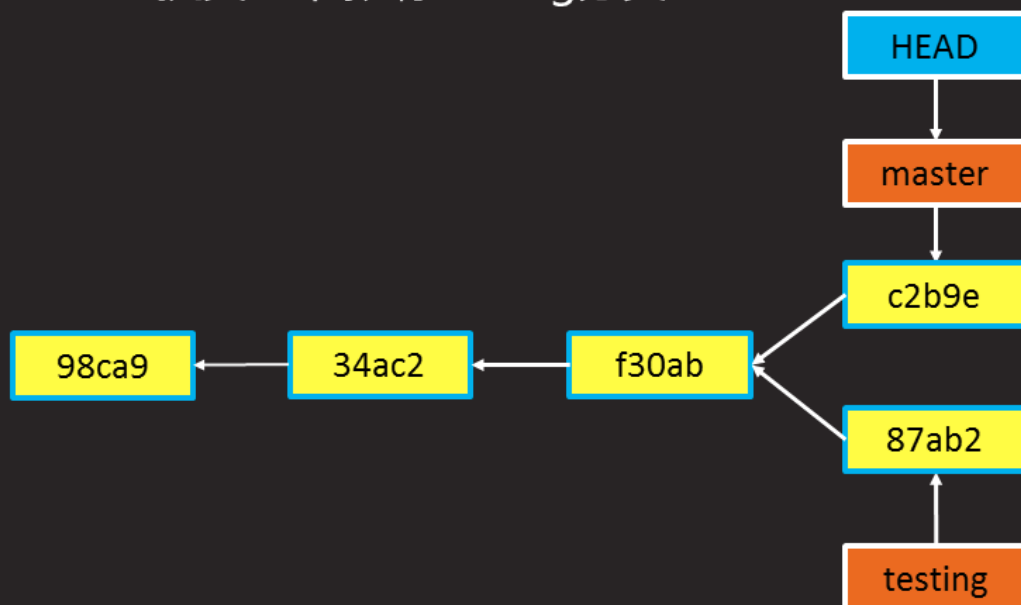
- 切换回master分支

知识讲解



多分支提交的流程（续3）

- master提交也不影响testing分支



知识讲解



案例3：Git分支操作

- 查看分支
- 创建分支
- 切换分支
- 合并分支
- 解决分支的冲突

课堂练习



Git服务器

SSH协议

- 密码认证访问
 - 服务器安装git
 - 使用git命令初始化版本仓库
 - 客户端使用SSH远程访问（可读写权限）

```
[root@svr5 ~]# git init /var/git --bare
```

```
[root@svr5 ~]# ls /var/git/
```

```
[root@client1 ~]# git clone root@服务器IP:/var/git  
正克隆到 'git'...
```

```
root@服务器IP's password: <需要输入密码>
```


SSH协议 (续1)

知识讲解

- 免密码远程Git服务器 (密钥授权)
 - 客户端生成SSH密钥
`[root@client1 git]# ssh-keygen -f /root/.ssh/id_rsa -N "`
 - 将密钥拷贝给Git服务器
`[root@client1 git]# ssh-copy-id Git服务器IP地址`
 - 测试上传、上传代码到远程服务器
`[root@client1 git]# git clone root@服务器IP:/var/git`
`[root@client1 git]# git push`



Git协议

知识讲解

- Git协议访问支持无授权访问 (只读)
 - 服务器安装git-daemon软件包
`[root@svr5 git]# yum -y install git-daemon`
 - 服务器初始化仓库 (必须要在/var/lib/git/目录建仓库)
`[root@svr5 git]# git init --bare /var/lib/git/project`
 - 服务器启动Git服务
`[root@svr5 git]# systemctl start git.socket`
 - 客户端使用git协议访问 (只读)
`[root@svr5 git]# git clone git://服务器IP/project`



HTTP协议

知识讲解

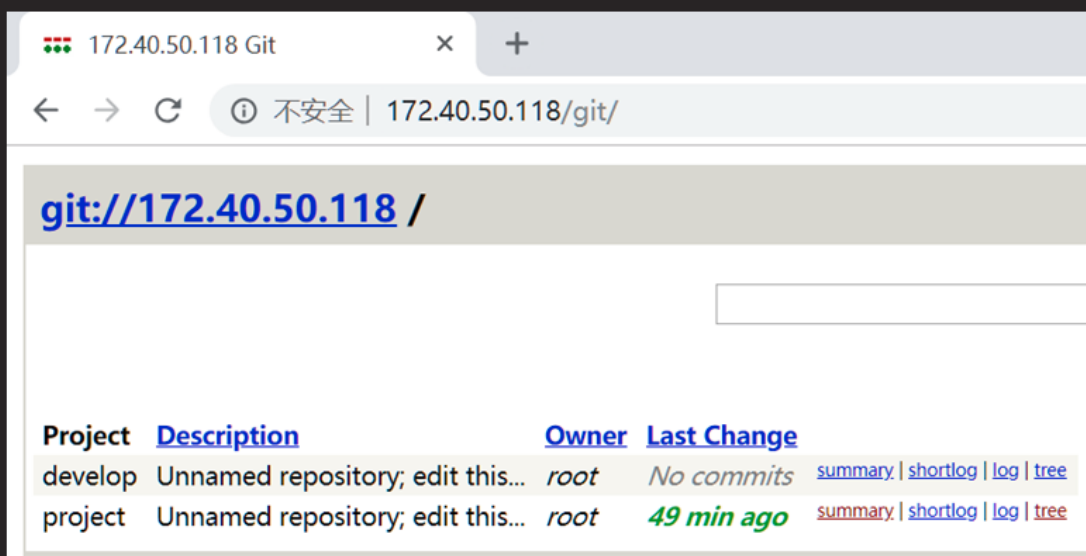
- 支持浏览器访问
 - 服务器安装gitweb软件包
`[root@svr5 git]# yum -y install httpd gitweb`
 - 修改配置文件，设置仓库根目录
`[root@svr5 git]# vim +11 /etc/gitweb.conf`
`$projectroot = "/var/lib/git";` #添加一行
 - 启动httpd服务
`[root@svr5 git]# systemctl start httpd`



HTTP协议（续1）

知识讲解

- 客户端使用浏览器访问



案例4：Git服务器

- SSH协议服务器
- Git协议服务器
- HTTP协议服务器

课堂练习



GitHub

简介

知识讲解

- GitHub是一个面向开源及私有软件项目的托管平台，因为只支持git 作为唯一的版本库格式进行托管，故名GitHub。
- GitHub于2008年4月10日正式上线



使用GitHub

知识讲解

- 注册、上传代码

Username

Email

Password

Use at least one letter, one numeral, and seven characters.

[Sign up for GitHub](#)

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We'll occasionally send you account related emails.

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

[Read the guide](#) [Start a project](#)



使用GitHub（续1）

- 注册、上传代码

知识讲解

Owner: redhatedu / Repository name: testaa ✓

Great repository names are short and memorable. Need inspiration?

Description (optional)

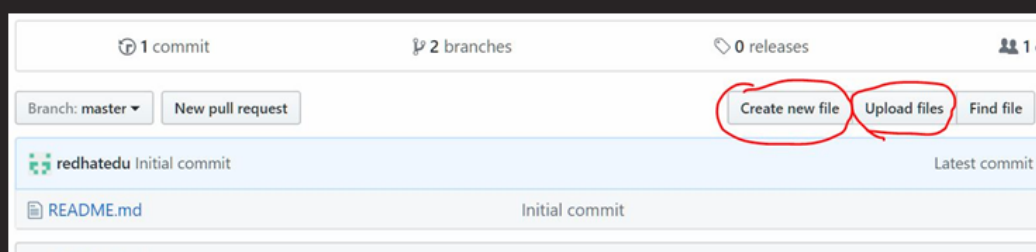
Public: Anyone can see this repository. You choose who can commit.

Private: You choose who can see and commit to this repository.

☒ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip

Add .gitignore: None Add a license: None ⓘ

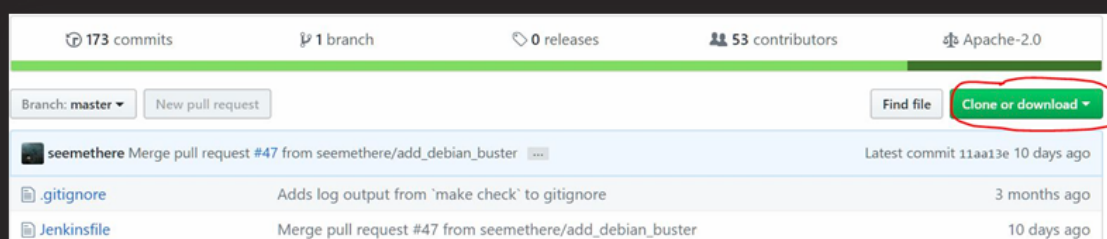
Create repository



使用GitHub（续2）

- 下载代码

知识讲解

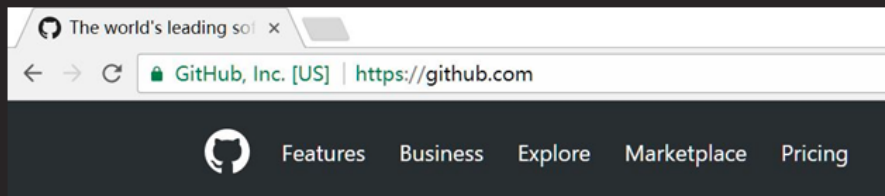


```
[root@test ~]# git clone https://github.com/docker/docker-install
Cloning into 'docker-install'...
remote: Counting objects: 493, done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 493 (delta 13), reused 13 (delta 6), pack-reused 467
Receiving objects: 100% (493/493), 184.10 KiB | 242.00 KiB/s, done.
Resolving deltas: 100% (207/207), done.
```

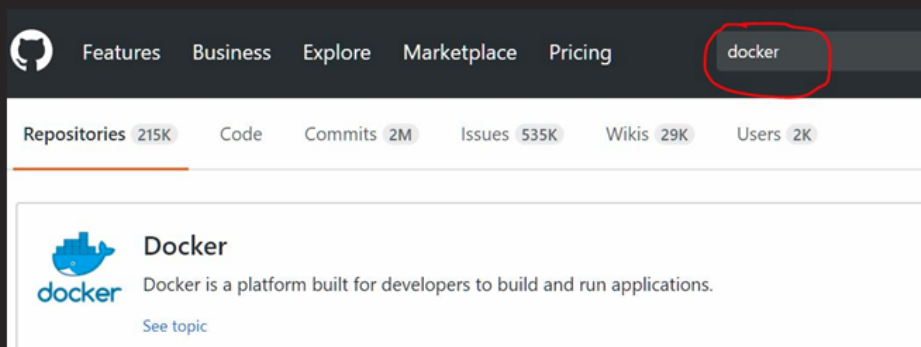


使用GitHub (续3)

- 登录网址 (<https://github.com/>)



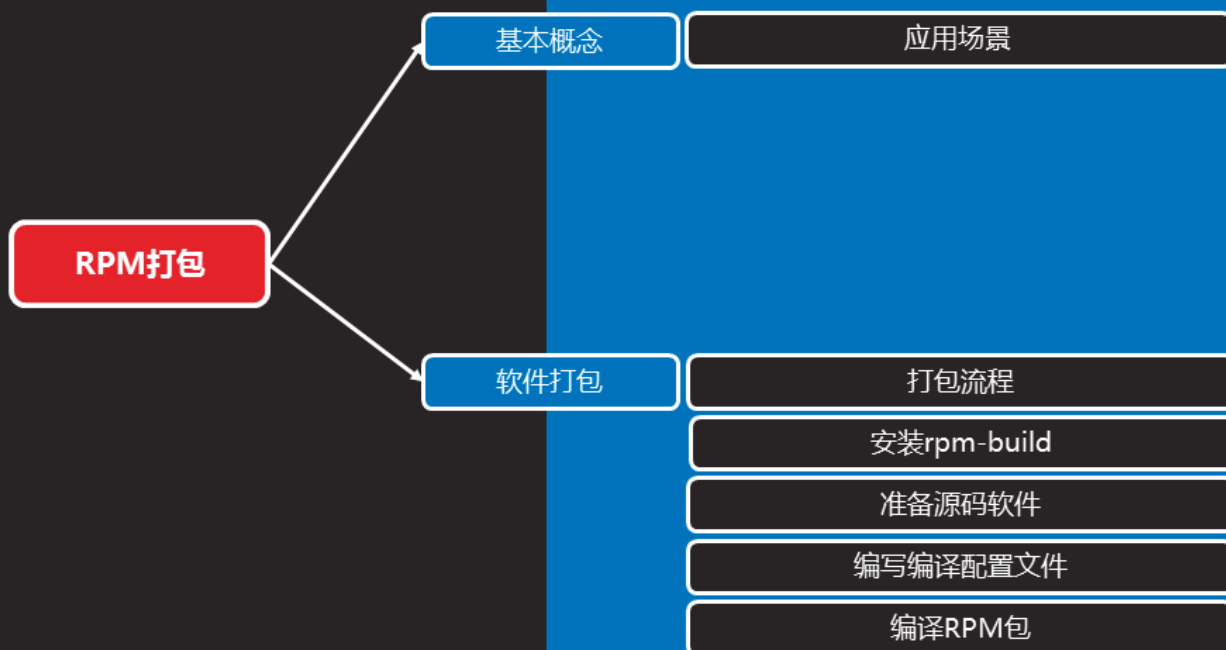
- 搜索项目



知识讲解



RPM打包



基本概念

应用场景

- 官方未提供RPM包
- 官方RPM无法自定义
- 大量源码包，希望提供统一的软件管理机制

软件打包

打包流程

- 准备源码软件
- 安装rpm-build
- 编写编译配置文件
- 编译RPM包

安装rpm-build

- 编译打包RPM的命令工具

```
[root@svr5 ~]# yum -y install rpm-build  
[root@svr5 ~]# rpmbuild -ba test.spec  
[root@svr5 ~]# ls
```

//生成rpmbuild目录

知识讲解



准备源码软件

- 将源码包复制到rpmbuild子目录

```
[root@svr5 ~]# cp nginx-1.12.2.tar.gz rpmbuild/SOURCES/
```

知识讲解



编写编译配置文件

- 新建SPEC文件

知识讲解

```
[root@svr5 ~]# vim /root/rpmbuild/SPECS/nginx.spec
```

```
Name:hello           //软件名称
Version:             //软件版本
Release: 1           //RPM版本
Summary:             //描述
Group:               //软件组
License:             //协议
URL:                 //网址
Source0:             //源码文件
BuildRoot:           %(mktemp -ud %[_tmppath]/%{name}-%{version}-%{release})
                      //临时编译目录
```



编写编译配置文件（续1）

- 新建SPEC文件

知识讲解

```
[root@svr5 ~]# vim /root/rpmbuild/SPECS/nginx.spec
```

```
... ..
BuildRequires:       //编译时依赖包
Requires:            //安装时依赖包
%description         //详细描述
%prep                //安装前准备，解压
%setup -q            //系统使用setup自动解压，安静模式
%build               //编译需要执行的命令
make
%configure           //配置时需要执行的命令
make %{?_smp_mflags}
%install              //安装时需要执行的指令
rm -rf %{buildroot}
make install DESTDIR=%{buildroot}
```



编写编译配置文件（续2）

- 新建SPEC文件

```
[root@svr5 ~]# vim /root/rpmbuild/SPECS/nginx.spec
```

```
... ..
```

```
%clean                                //清理时需要执行的指令
```

```
rm -rf %{buildroot}
```

```
%files                                //定义打包文件列表
```

```
%defattr(-,root,root,-)
```

```
%doc
```

```
%changelog                             //软件修改历史
```

知识讲解



编译RPM包

- 使用spec文件编译RPM包

```
[root@svr5 ~]# rpmbuild -ba /root/rpmbuild/SPECS/nginx.spec
```

- 安装测试RPM包

```
[root@svr5 ~]# rpm -qpi XXX.rpm
```

```
[root@svr5 ~]# rpm -qpl XXX.rpm
```

```
[root@svr5 ~]# rpm -ivh XXX.rpm
```

知识讲解



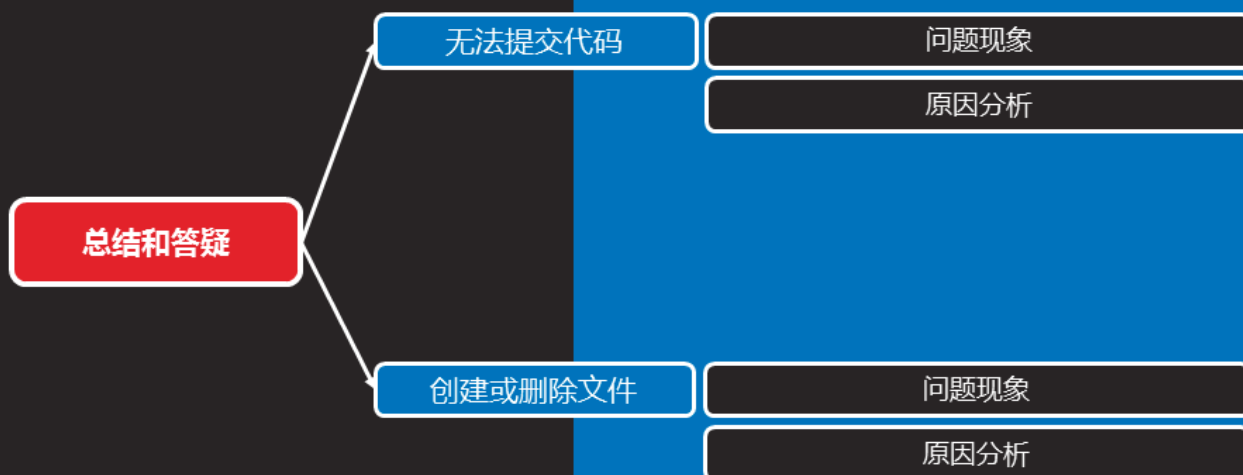
案例5：制作nginx的RPM包

课堂练习

- 使用nginx源码制作对应的RPM包
 - nginx源码版本为1.12.2
 - 编写spec配置文件
 - 使用rpmbuild工具进行rpm打包
- 使用RPM包安装nginx软件



总结和答疑



无法提交代码

问题现象

- 故障错误信息

```
[root@client~]# git add .  
fatal: Not a git repository (or any of the parent directories): .git
```

原因分析

知识讲解

- 分析故障信息
 - fatal: Not a git repository (or any of the parent directories): .git
- 分析故障原因
 - 没有进入代码仓库就执行了add指令
 - Git指令必须在git仓库中操作



创建或删除文件

问题现象

- 故障错误信息

```
[root@svr5 ~]# rm a.txt  
[root@svr5 ~]# mkdir test; touch test.txt  
[root@svr5 ~]# svn commit -m "aa"
```

知识讲解



原因分析

- 分析故障信息
 - 对版本库中的代码不可以直接使用命令删除或创建
- 分析故障原因
 - git add才可以让git识别到新加的文件
 - git rm才可以删除文件

知识讲解



