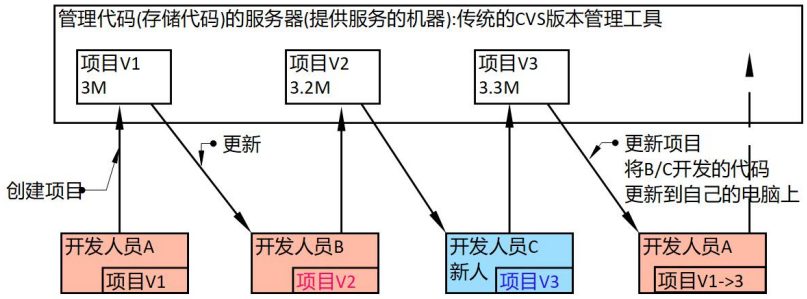
第二十讲、Linux 服务配置与版本控制

课程大纲	课时	内容	目标	重点	作业
常见代码托管平台	1	常用的代码托管平台 码云，GitLab	了解常见的代码托管平台	☆	整理 Linux 基础相关知识点
git、GitHub	3	git 基本使用	掌握 git 基本命令	★	
协同开发	1	基于 Python 环境、git 版本控制下的多人协同开发	了解协同开发流程	☆	
Linux 常见服务	1	ssh 服务的安装和使用	掌握 ssh 的使用	★	
作业讲解	2	整理 Linux 基础相关知识点			

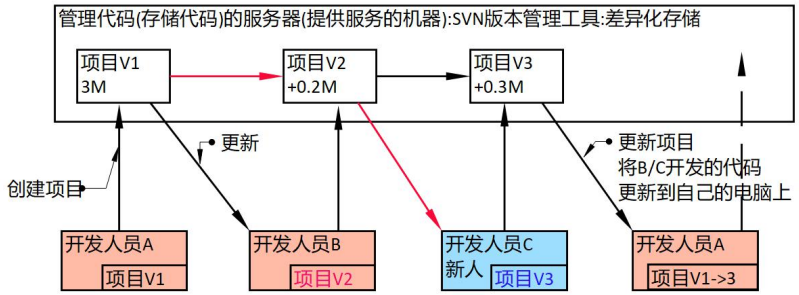
一、GIT 版本管理

项目中的多人协同开发，涉及到代码的质量管理、提交认证、代码合并、错误排查等等一系列问题，如果通过人工操作的方式会造成非常大的工作负荷，通常这类功能我们会使用业内相对权威的公司或者机构、社区管理的第三方代码托管平台完成代码的在线托管，通过对应的操作管理工具如 Git、SVN 等完成对托管项目的管理和维护工作。

- CVS 代码版本管理工具
  - 解决了多个人协同开发，代码合并的问题：从服务器更新代码完成合并
  - 解决了多人协同开发，排查问题的难度：每次提交都有记录
  - 新的问题：代码在服务器上同时存储了多个版本(完整)，重复量过大
    - 对于服务器的要求(存储空间)更高



- SVN 代码版本管理工具
  - 优点：在实现了 CVS 所有功能的基础上，实现了差异化管理，让服务器的存储空间不再出现重复存储代码的情况，极大的节省了服务器空间！



✧ GIT 代码版本管理工具

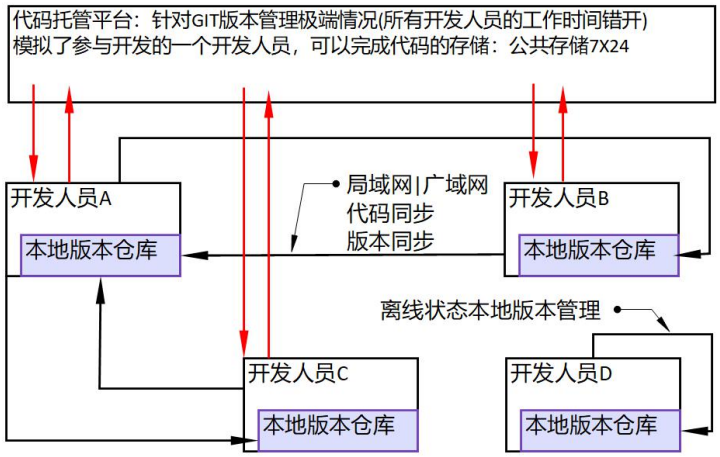
相对于传统的 SVN 管理工具而言

✧ 去服务器，没有服务器的概念，每个开发人员的本地仓库及时开发端也是服务器

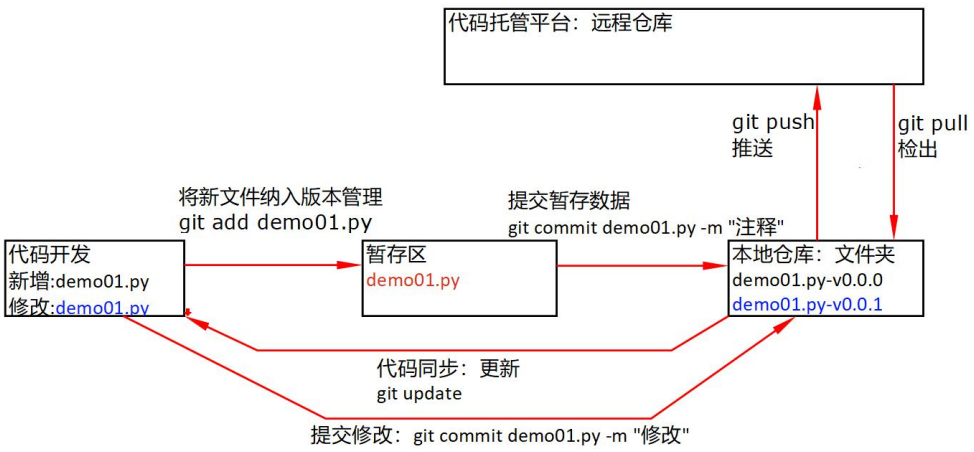
■ 可以和另一个开发人员之间，完成代码同步，完成版本管理

✧ 基于网络，突破了局域网的限制，可以让开发人员随时随地在任何时间办公，可以通过广域网完成代码的同步和版本的管理

✧ 基本本地，在没有网络的情况下，可以让代码在自己本地的仓库中完成代码的同步和管本的管理，将不同功能不同批次的代码管理在不同的版本中



✧ 为什么 GIT 可以基于网络、可以基于本地完成版本管理？



（一）GITHUB

GitHub 是目前企业项目开发在线托管最大的平台之一，访问官方网站并注册自己的开发者账号：
GITHUB 的注册和仓库创建参考《GITHUB》.doc 文档

（二）GIT

Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。是一个开源的分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目。Git 与常用的版本控制工具 CVS,Subversion 等不同，它采用了分布式版本库的方式，不必服务器端软件支持。

访问 Git 官方网站 https://git-scm.com/下载适用于自己操作系统平台的 Git 软件：

将下载好的 Git 安装到自己的操作系统中，鼠标右键菜单中就能看到 Git 相关操作的两个选项

Git GUI Here

Git Bash Here

此时就可以在计算机上直接使用 Git 的界面操作或者命令行完成代码和远程仓库之间的管理和维护了。

（1） Git 和 SVN 的区别

Git 与 SVN 区别点：

GIT 是分布式的，SVN 不是

（2） 工作流程

git 版本管理工具，是通过分布式的代码版本管理方式，没有服务端和客户端之分，每个开发人员即是服务端也是客户端，这样将每个开发人员都独立出来，既方便了多人在线时的的代码同步，也保证了离线状态下的版本提交和维护管理，协作过程如下图所示：

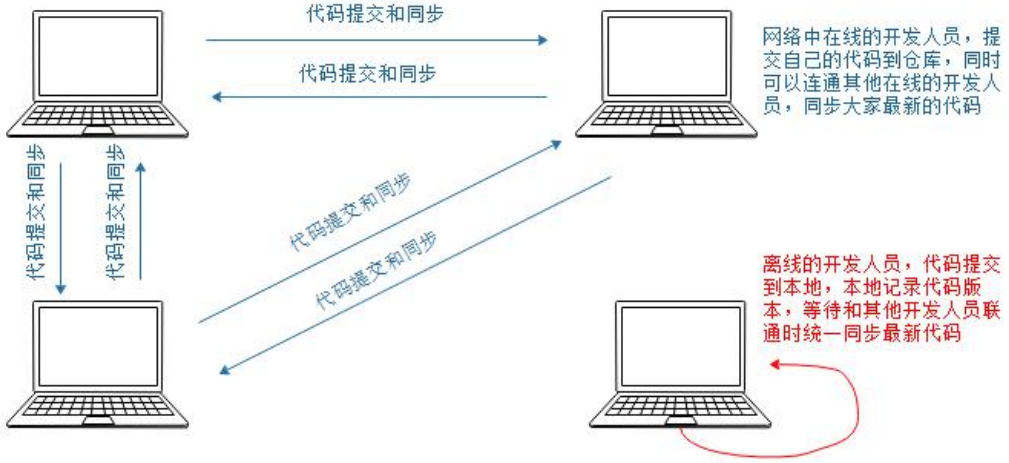


图 20-1

将代码区分在了不同的区域，以方便同步多个开发人员之间的代码协同工作，主要区分为三个部分：

- 暂存区：临时存储代码的缓存区域，暂存区的存在，是为了将非常频繁的小改动统一存储起来，等开发人员认为自己的修改已经成型可以提交了，在将暂存区的数据提交到本地仓库
- 本地仓库：本地仓库是一个接受代码版本管理的残酷，这个仓库中数据会按照提交历史进行版本管理控制，方便了开发人员在没有网络连通的情况下提交的数据依然接受版本管理
- 远程仓库：远程仓库是一个多人协同开发时的公共仓库，保持 24 小时在线，这样能最大程度的保证所有开发人员都能在联网的第一时间同步到最新的代码

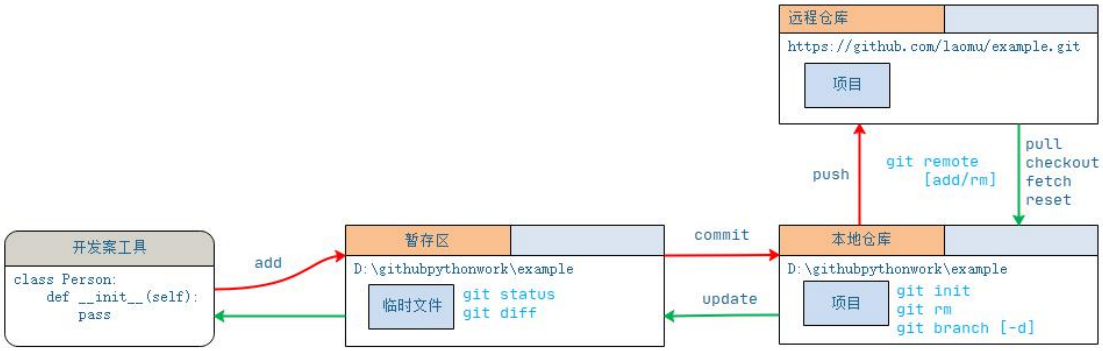


图 20-2

（3） 仓库初始化：git init

本地仓库，是在当前计算机中完成代码版本的管理和维护的文件夹，将一个本地存在的普通的文件夹通过 init 初始化为 git 本地仓库。

init 执行完成后，会在当前文件夹中生成一个隐藏文件夹.git/，文件夹中初始化了 git 仓库基本信息，以及记录本地版本管理的所有数据

（4） Git 配置

本机完成 git 软件安装之后，需要配置用户的基本信息，方便在后续进行代码版本管理和维护。

Git init 初始化一个 git 目录

创建 git 项目提交身份

配置局部身份

git config user.name laobian 设置提交名

git config user.email laobian@qq.com 设置提交邮箱

cat .git/config 查看局部提交身份

配置局部身份

Git config --global user.name laobian 设置全局提交名

Git config --global user.email laobian@qq.com 设置全局提交邮箱

查看全局配置

Git ~/.gitconfig

优先级

就近原则，有局部用局部，没有局部用全局，二者不可以都没有

查看当前版本的状态

（5） 仓库同步

仓库同步，就是将本地仓库和远程仓库之间建立关关系，我们就可以通过这样的关联关系完成本地仓库和远程仓库之间的代码的同步和版本的管理维护。

而和远程仓库之间的同步，有两种方式是比较容易混淆的：

- git clone
  - 克隆项目，在知晓远程仓库地址 https/ssh 链接的情况下，可以通过 git clone 命令将项目从远程仓库同步到本地仓库
  - 克隆项目是只读模式，只能读取仓库中的代码数据，不能修改仓库中的代码，也不能向仓库提交新的代码数据
- git remote add / push / pull

  git remote add 命令是用于将本地仓库和远程仓库建立关联关系，有了关联关系是进行代码维护的前提条件，仓库中对该用户进行授权之后，就可以通过本地仓库完成远程仓库的代码维护了

  git push 代码维护过程中，推送本地仓库的数据到远程仓库

  git pull 代码维护过程中，拉取远程仓库的数据到本地仓库

本地文件夹和远程仓库同步关联，本地文件夹中创建 readme.txt 文件

完成本地仓库和远程仓库的同步

$ git add readme.txt

$ git commit -m "初始化"

生成 SSH 秘钥 key 并配置到 github 中

$ ssh-keygen -t rsa

打开自动生成的文件/c/Users/DAMU/.ssh/id_rsa，将公钥文件内容，添加到 github 上

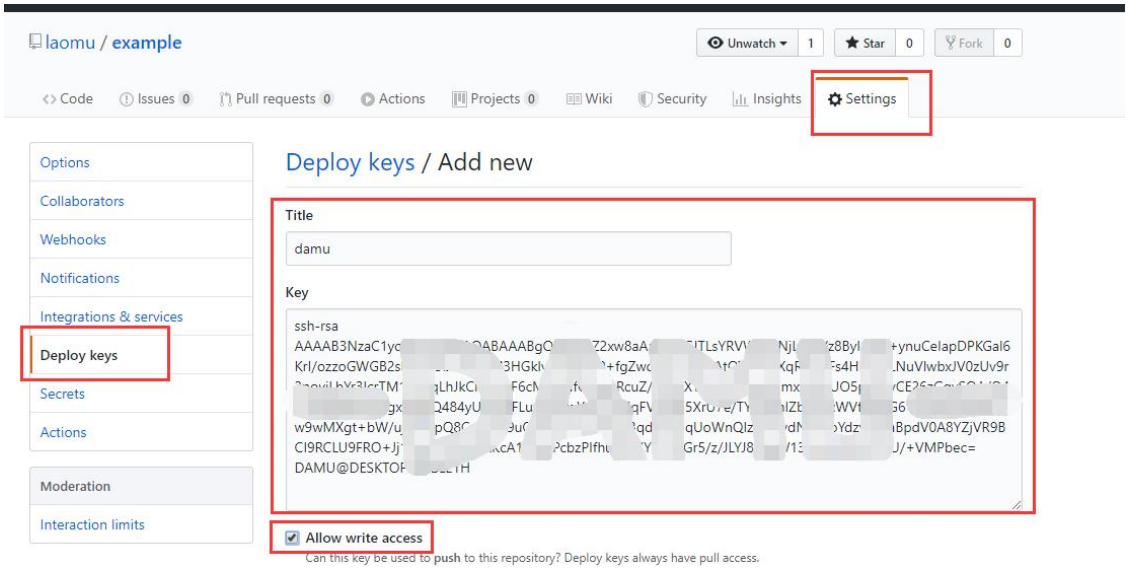


图 20-3

此时本地 commit 的文件，就可以正常 push 到远程仓库了，需要注意~此时提交本地仓库数

据到远程仓库，需要当前用户先登录自己的账号，才能正确提交数据
此时刷新网页查看远程仓库，本地提交的文件已经出现在远程仓库中了
（6） 添加快照：git add
（7） 状态检查：git status
（8） 代码提交：git commit
（9） 差异排查：git diff
（10） 历史记录：git log
如果提交版本信息过多，可以只查看前 n 条信息
git log -n 2
git log --after='2020-2-4 9:55:00'
如果协同开发人数较多，可以指定查看某个作者提交的版本信息
git log --author="laomu" -n 2
（11） 回滚：git reset HEAD
（12） 分支操作：git branch
创建分支：git branch <branch>
查看分支：git branch
切换分支：git checkout <branch>
删除分支：git branch -d <branch>
（13） 合并分支：git merge
（14） 远程仓库添加：git remote
（15） 远程代码提取：git pull
（16） 远程仓库推送：git push

（三）协同开发

有了 git 版本管理工具，代码的版本控制和维护就会变得非常简单，但是本地代码开发一般使用的都是 IDE 工具，高端工具的使用如果再结合频繁的命令行操作，无形中会增加很多工作量，所以多人协同开发模式下，git 和工具的结合变得非常重要。

1、 GitHub 远程配置
公司的远程仓库一般都搭建在 github 上，或者代码的安全级别较高、同时项目组经费充裕的情况下，会购买专门的服务器搭建公司的私有服务器。
我们这里以 Github 为例，说明多人协同开发模式下，远程仓库的开发配置，通常配置步骤如下：

- 收集开发人员计算机上的 ssh 公钥
- 添加开发人员公钥到指定的仓库

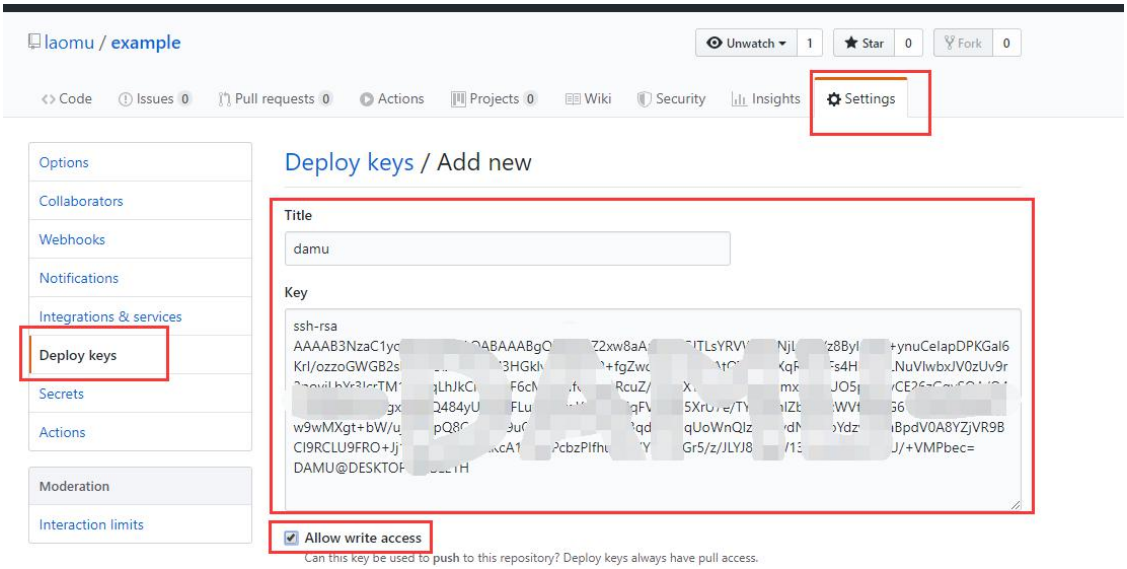


图 20-4

（2） Pycharm 本地配置

Pycharm 是目前 Python 应用开发的主要工具之一，我们以它为例说明 git 和高级工具的集成方式，同时大家如果在使用其他的开发工具的话，操作的思路和配置的方式都是类似的，通常的配置步骤如下：

- 工具配置执行选项：git

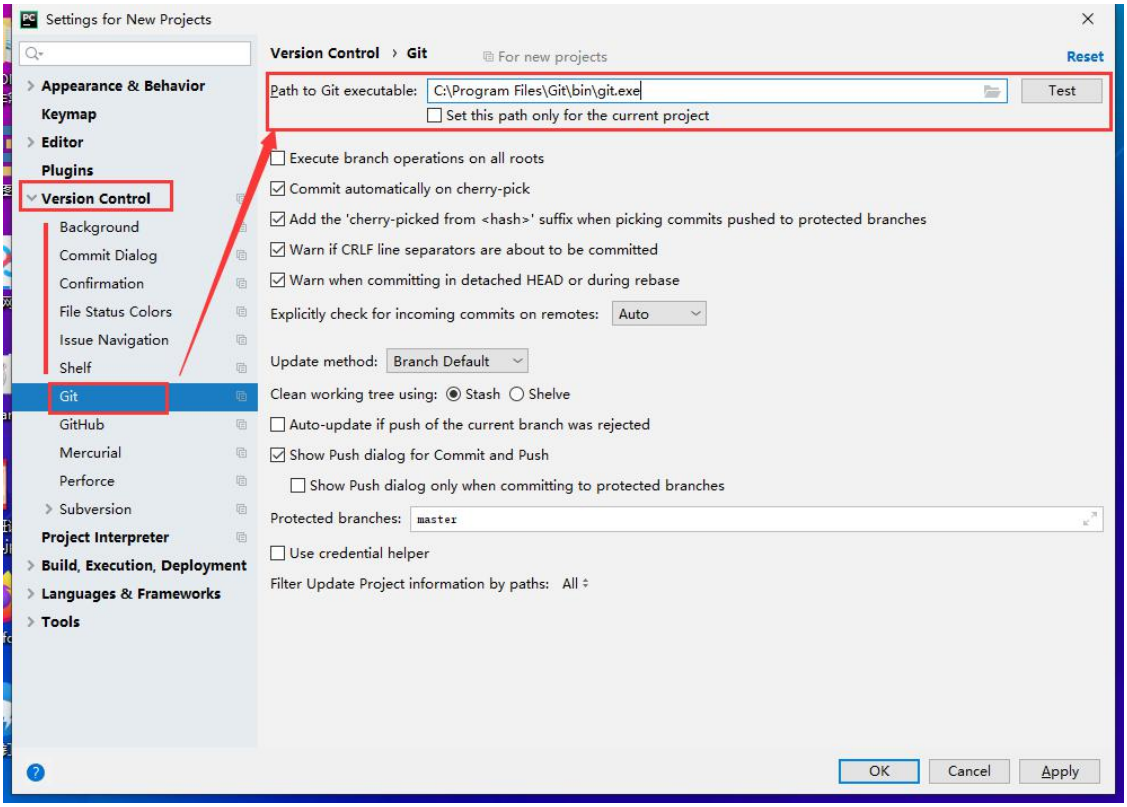


图 20-5

- 工具配置登录选项：github

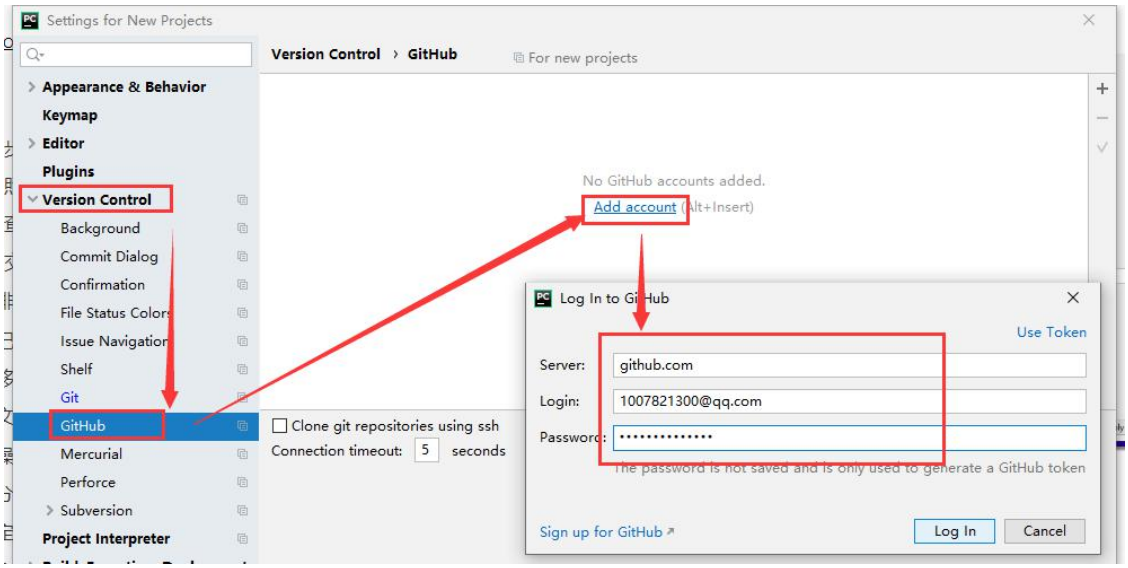


图 20-6

- 工具配置和远程仓库的关联
  - –配置从远程仓库检出项目：vcs->checkout
  - –配置推送本地项目到远程仓库

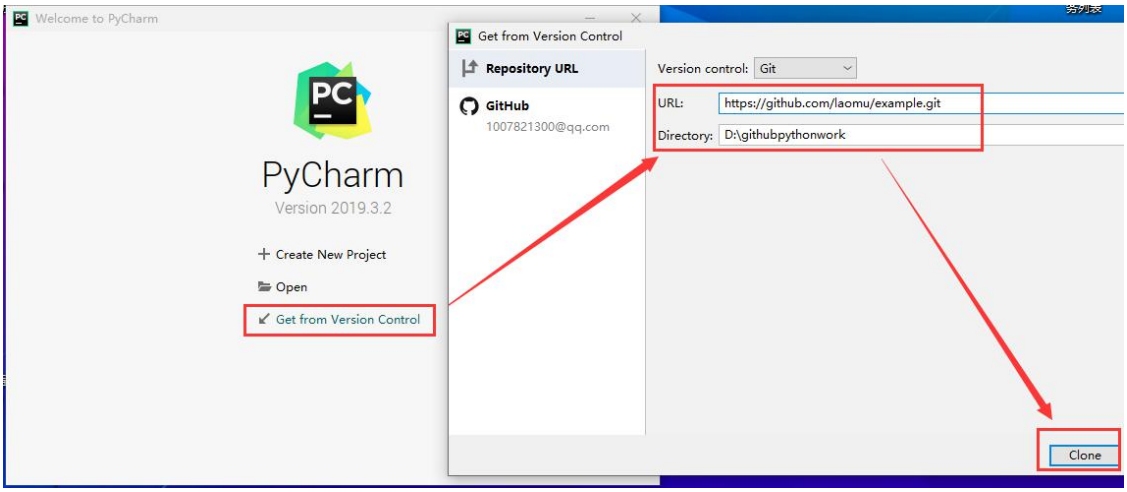


图 20-7

- 工具配置启用 git control

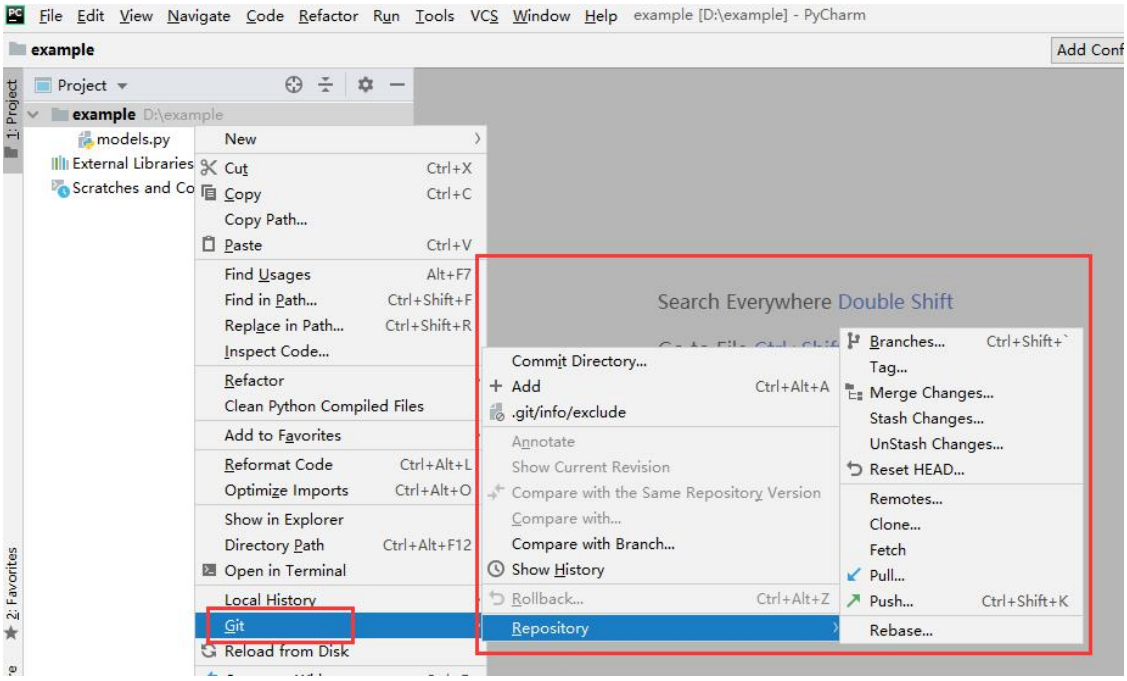


图 20-8

二、SSH 服务

（一）SSH 概述

SSH 是 Secure Shell Protocol 的简写，由 IETF 网络工作小组（Network Working Group)制定；在进行数据传输之前，SSH 先对联机数据包通过加密技术进行加密处理，加密后在进行数据传输。确保了传递的数据安全。
SSH 是专为远程登录会话和其他网络服务提供的安全性协议。

（二）SSH 服务端

SSH 服务端是一个守护进程 (daemon).他在后台运行并响应来自客户端的连接请求。 SSH 服务端的进程名为 sshd，负责实时监听远程 SSH 客户端的远程连接请求，并进行处理，一般包括公共密钥认证、密钥交换、对称密钥加密和非安全连接等。这个 SSH 服务就是我们前面基础系统优化中保留开机自启动的服务之。
ssh 客户端包含 ssh 以及像 scp(远程拷贝） slogin(远程登陆) sftp(安全 FTP 文件传输）等应用程序。

CentOS 默认是安装好 SSH 的，检测 SSHD 服务是否已经启动

```
systemctl status sshd
```

CentOS 7 启动|停止|重启 以及 设置开机启动 SSHD 服务

```
systemctl start sshd       # 启动 sshd 服务
```

```
systemctl restart sshd        # 重启 sshd 服务
systemctl stop sshd           # 停止 sshd 服务
systemctl enabled sshd        # 设置开机启动 sshd 服务
```

（三）SSH 客户端

通常在系统中已经集成了 SSH 客户端，如果检测没有 SSH 客户端可以安装对应的图形界面软件如 SecureCRT、Xshell、Termius 等，也可以直接安装命令行插件。这里以 Xshell 为例进行讲解。

（1） 配置服务端

服务端 SSHD 配置文件位于/etc/sshd/sshd_config，修改如下配置允许 root 用户远程连接

注意：生产环境严禁允许 root 远程登录

PermitRootLogin yes

（2） 客户端安装

官方网站下载 Xshell 并安装在系统中，软件开始打开界面如图

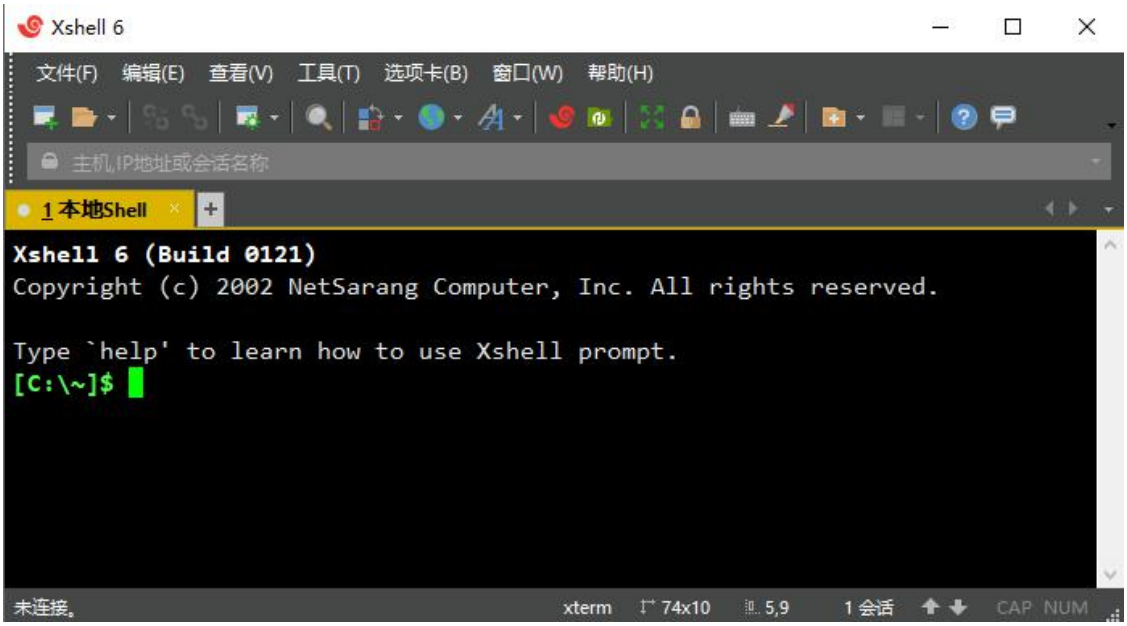


图 20-9

（3） SSH 远程连接

ALT+N 新建连接会话，输入远程 Linux 服务器 IP 地址

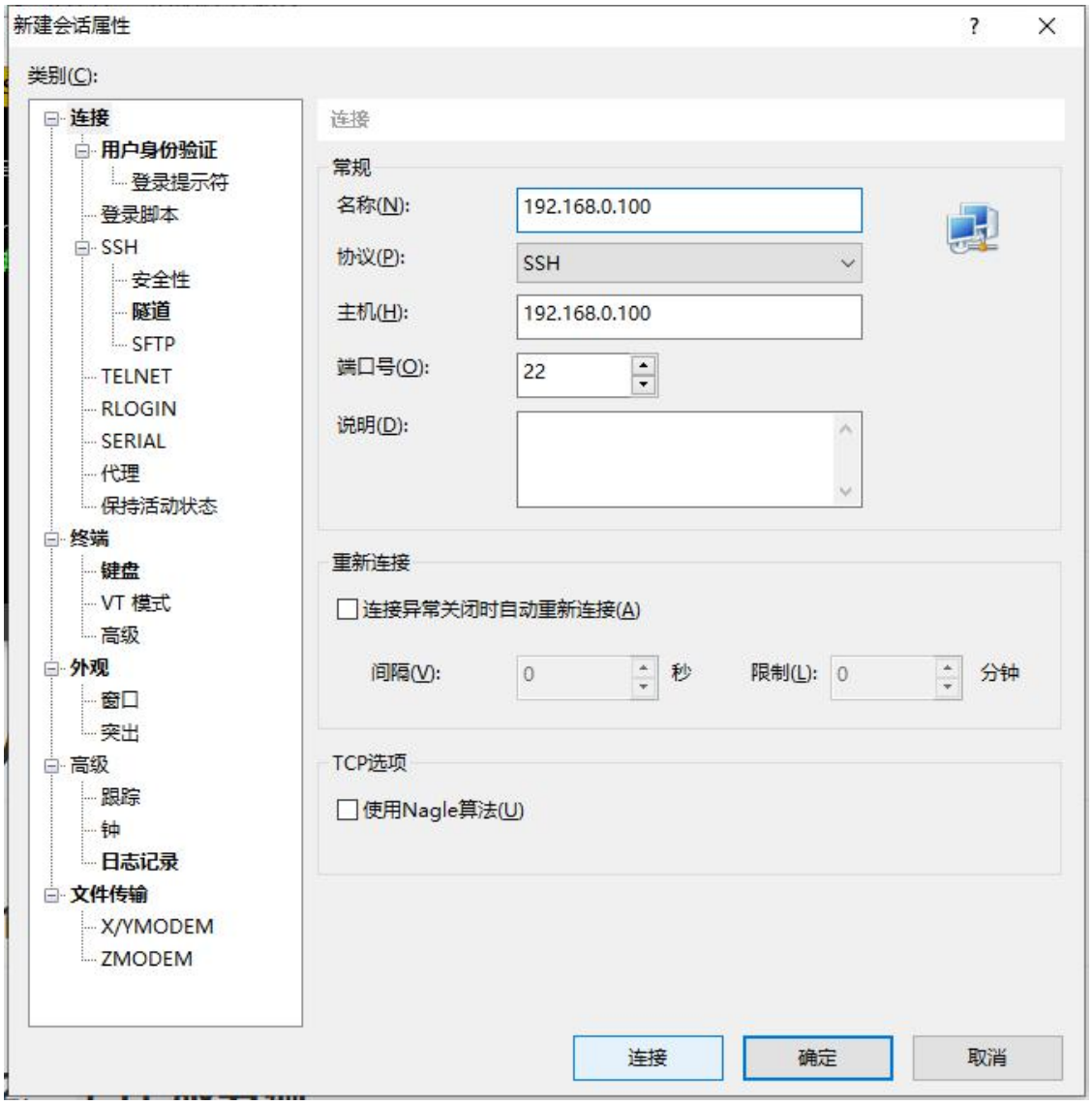


图 20-10

点击确定，输入连接账号：root

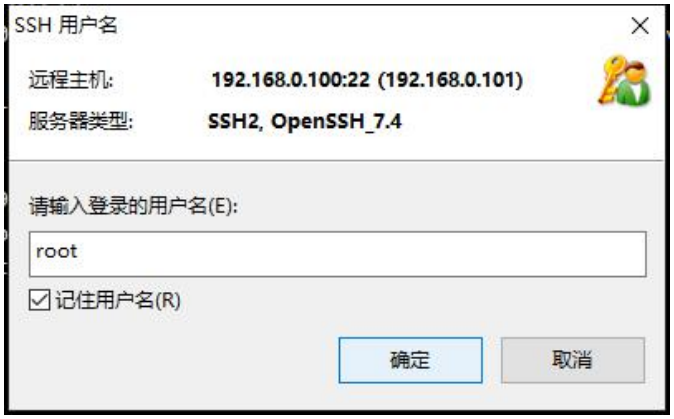


图 20-11

点击确定，在下一个窗口中输入密码，并勾选记住密码

图 20-12

点击确定后，如果弹出连接确认窗口点击接受即可。最后软件完成远程登录的窗口：

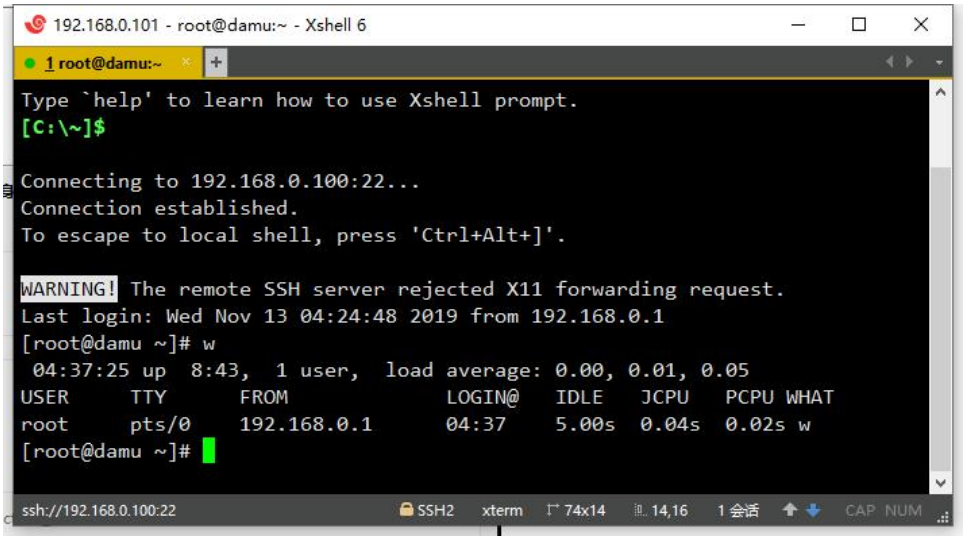


图 20-13

（四）免密登录

软件中可以选择记住密码来完成免密登录，如果是在命令行，我们可以通过公布共享公钥的方式配置免密登录。

（1） 服务端配置

SSH 服务端生成 ssh_id，执行命令创建：

ssh-keygen：创建 ssh.id 命令

-t：指定算法，默认 rsa
-C：注释，一般添加作者邮箱
ssh-keygen -t rsa -C "********@qq.com"

确认客户端 IP 地址:192.168.12.132，将公钥发布到指定 IP 地址客户端
确认客户端已经安装过 SSH 服务
确认客户端和服务端 ping 通信通过
ssh-copy-id <ip>
（2） 客户端连接
客户端直接通过 ssh <ip>命令连接服务器
基本操作语法：
ssh $user:$pass@$ip
免密登录
ssh $ip

作业：

整理 Linux 基础相关知识点