Git和代码托管中心

代码托管中心的作用：维护远程库

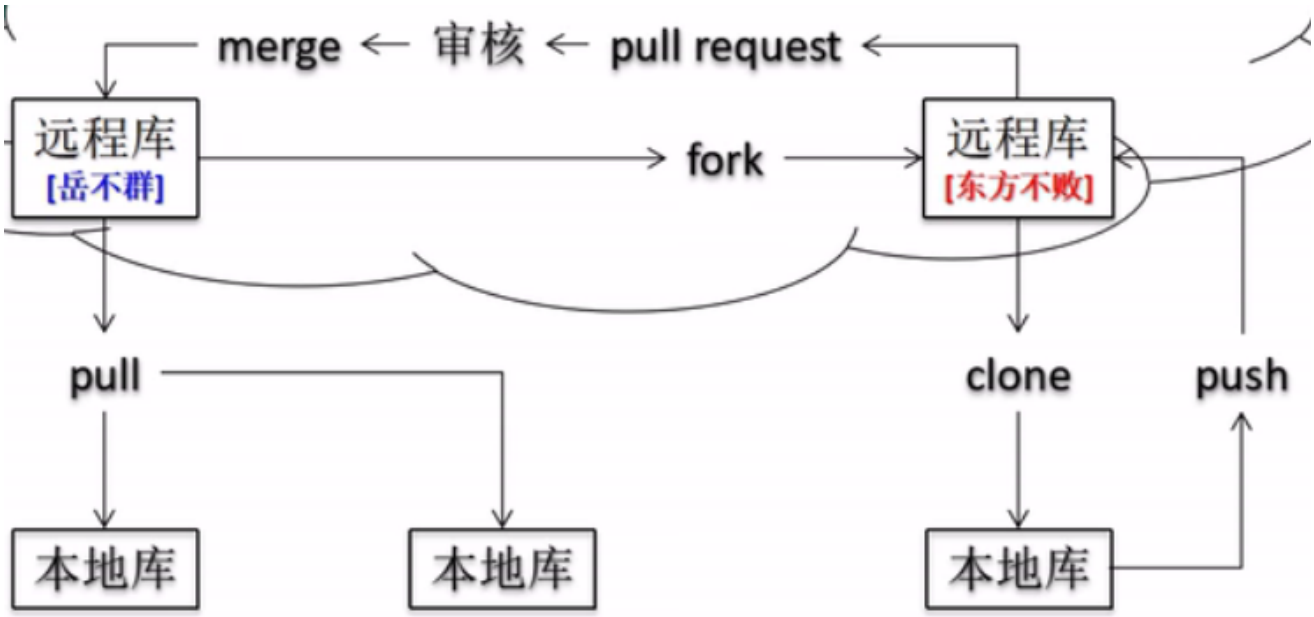
- 局域网下：Gitlab
- 外网环境下：Github、码云

本地库和远程库交互方式

- 团队内部协作

```
wangjian10@wangjian10:~/Github_blog$ git remote -v
origin  https://github.com/bestxiaojian/Blog.git (fetch)
origin  https://github.com/bestxiaojian/Blog.git (push)
```

- 跨团队协作



本地库操作

- **查看本地库提交记录**

  除了git log常规查看，还可以使用:

  git log --pretty=oneline 只显示在一行或者**git log --oneline**或者**git reflog** :显示到某个版本，HEAD需要移动几步

- **本地库记录的前进后退操作**：有如下3种方式
  - 基于索引值操作（**推荐使用索引值方式**）

    git reset (--hard) + commit id(hash code)
  - 使用^符号：**只能往后退版本**

比如：**git reset --hard HEAD^^**，^的数量代表往后退的步数

- 使用~符号

git reset --hard HEAD~+数字：比如 git reset --hard HEAD~4 往当前版本后退4步

- git reset 命令三个参数对比
  1. --soft 参数：仅仅在本地库移动HEAD指针，暂存区和工作区不变（“会显得暂存区和工作区很新”）
  2. --mixed参数：在本地库移动HEAD指针，同时改变暂存区（“会显得工作区很新”）
  3. --hard参数：在本地库移动HEAD指针，暂存区和工作区也改变

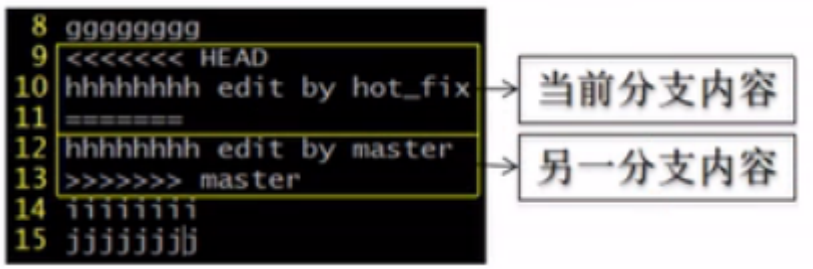
- **删除文件的找回**

  前提：删除前，**文件存在时的状态提交到了本地库**，否则不能找回

  操作： 还是git reset --hard + commit id（某个版本）

- **比较文件差异**
  - **git diff** + [文件名]：将工作区中的文件和暂存区进行比较
  - **git diff** + [本地库中某个历史版本] + [文件名]： 将工作区中的文件和本地库中对应的版本进行比较
- **分支操作**：这里主要提一下如下几个操作
  - 创建分支： git branch + 分支名
  - 合并分支：
    1. 切换到接受修改的分支上（即需要合入修改的分支） **git checkout** + 分支名
    2. 执行git merge + 分支名（含有修改的分支名）
    - **解决冲突**

      冲突的产生：不同分支修改同一文件的同一行时，产生冲突



      冲突解决：很简单，**解决上述冲突的文件，然后保存，然后依次执行git add 、git commit** 即可（git也有相应的提示）

远程库操作

- 创建远程库地址别名：方便后续操作，不用总记住或复制远程库地址

git remote add + 别名（比如origin) + 远程库地址,以后origin即代表远程库地址

```
wangjian10@wangjian10:~/Github_blog$ git remote add origin https://github.com/bestxiaojian/Blog.git
```

- 查看创建的别名：**git remote -v**

```
wangjian10@wangjian10:~/Github_blog$ git remote -v
origin  https://github.com/bestxiaojian/Blog.git (fetch)
origin  https://github.com/bestxiaojian/Blog.git (push)
```

- **推送操作**： **git push**

  git push origin master（远程分支名）

- 克隆操作：git clone

git clone + 远程仓库地址

效果如下：

1. 完整的把远程库下载到本地
2. 地址别名设置也克隆了下来，不用再设置（git remote -v）
3. 本地库已经初始化

- 远程库修改拉取（同步）到本地：**pull** = fetch + merge

git pull origin(远程库地址) master(远程分支名)

- **设置用户签名**

1. 作用：区分不同的开发人员

签名分为如下两类：

1. 项目级别\仓库级别：仅在当前本地库范围内有效
   - git config user.name + 用户名
   - git config user.email + email地址

   设置后，设置信息保存在当前git仓库的.git/config文件的[user]字段中
2. 系统用户级别：登录当前操作系统的用户范围
   - git config **--global** user.name + 用户名
   - git config **--global** user.email + 邮件地址

   设置后，设置信息保存在~/.gitconfig文件的[user]字段中

   通常只设置系统级别就可以了