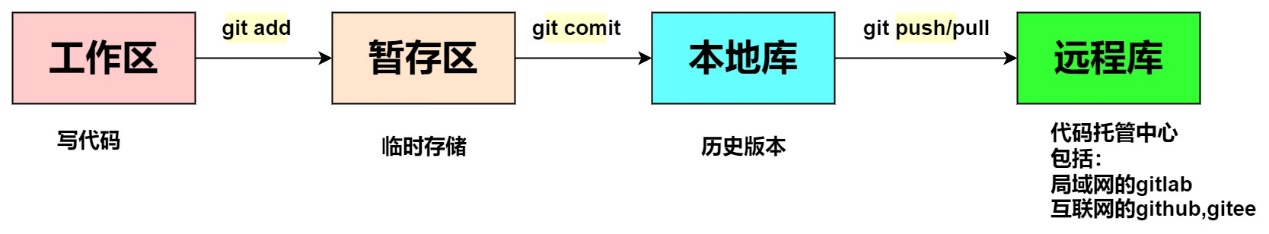
**软件工程中级实训 ——Git学习笔记：**

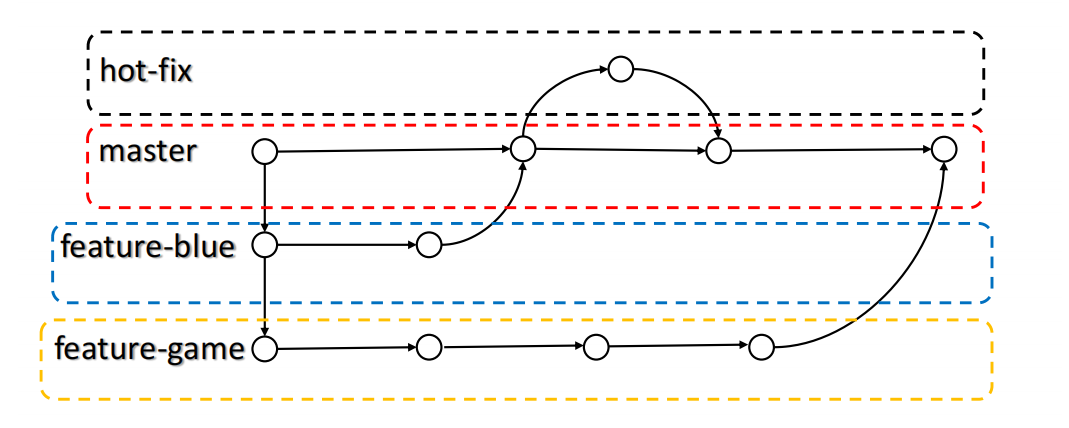
1. **Git概述：**
   1. Git是一个免费的、开源的分布式版本控制系统，可以快速高效地处理从小型到大型的各种项目。Git易于学习，占地面积小，性能极快。它具有廉价的本地库，方便的暂存区域和多个工作流分支等特性。经查阅资料，其性能优于大多数的版本控制工具。
   2. 需要版本控制的原因是从个人工作过渡到团队协作，版本控制最重要的是可以记录文件修改历史记录，从而使得用户用户能够查看历史版本，方便版本之间的切换。
2. **Git工作机制：**
   1. **Git的四部分**



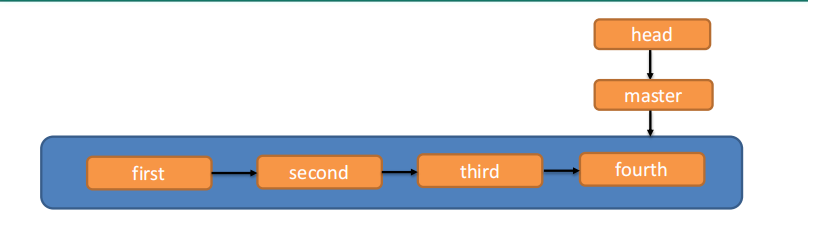
* 1. **Git的常用命令**

|  |  |
| --- | --- |
| **命令名称** | **作用** |
| git config --global user.name 用户名 | 设置用户签名 |
| git config --global user.email 邮箱 | 设置用户签名 |
| git init | 初始化本地库 |
| git status | 查看本地库状态 |
| git add 文件名 | 添加到暂存区 |
| git commit -m "日志信息" 文件名 | 提交到本地库 |
| git reflog | 查看历史记录 |
| git reset --hard 版本号 | 版本穿梭 |
| git branch 分支名 | 创建分支 |
| git branch -v | 查看分支 |
| git checkout 分支名 | 切换分支 |
| git merge 分支名 | 把指定的分支合并到当前分支上 |
| git remote -v | 查看当前所有远程地址别名 |
| git remote add 别名 远程地址 | 起别名 |
| git push 别名 分支 | 推送本地分支上的内容到远程仓库 |
| git clone 远程地址 | 将远程仓库的内容克隆到本地 |
| git pull 远程库地址别名 远程分支名 | 将远程仓库对于分支最新内容拉下来后与当前本地分支直接合并 |

* 1. **Git的分支**
     1. 在版本控制过程中，同时推进多个任务，为每个任务，我们就可以创建每个任务的单独分支。使用分支意味着程序员可以把自己的工作从开发主线上分离开来，开发自己分支的时候，不会影响主线分支的运行。对于初学而言，分支可以简单理解为副本，一个分支就是一个单独的副本。（分支底层其实也是指针的引用）



* + 1. **创建分支与切换分支**



对于之前创建的多个分支，如hot-master，master等都是指向具体版本记录的指针。而当前所在分支是由head决定的。所以创建分支的本质就是多创建一个指针：HEAD 如果指向 master，那么我们现在就在 master分支上HEAD 如果执行hotfix，那么我们现在就在hotfix分支上。也就是说切换分支的本质是移动head指针。

* + 1. **合并冲突：**

冲突产生的原因：

合并分支时，两个分支在**同一个文件的同一个位置**有两套完全不同的修改。Git无法替我们决定使用哪一个。必须**人为决定**新代码内容。

采取的措施是编辑有冲突的文件，删除特殊符号，决定要使用的内容。

* 1. **在github或者gitee上的操作：**
     1. **团队内协作和团队间协作**
        1. 团队内协作：不同团队成员之间pull,push,clone等操作进行版本的更替
        2. 团队间协作：另一个团队通过fork到自己的远程库，然后在本地进行修改之后，上传到自己的远程库，然后进行PullRequest，原先团队进行审核，通过后对不同代码的版本进行merge

1. **IDEA集成Git**
   1. **【可选】配置Git忽略文件：**为什么要忽略：与项目的实际功能无关，不参与服务器上部署运行，忽略掉他们能够屏蔽IDE工具之间的差异。
   2. **对于IDEA而言，操作Git只需要点击即可实现命令行的操作：**
      1. 在IDEA上面登录自己的Gitee账号，然后根据相应的位置点击即可实现上述命令行功能：

|  |  |
| --- | --- |
|  |  |

* + 1. 同时也可以在不同版本之间和不同分支之间进行切换，本组成员部分使用效果如下：

|  |
| --- |
|  |
|  |