Git简单使用心得

1. Commit
   1. 每次提交产生一个Commit对象，每个Commit对象有一个与之对应的CommitId
2. 分支与HEAD
   1. 分支指向某个CommitId
   2. HEAD指向当前分支
3. 分支
   1. 本地分支
      1. 创建分支
         1. git branch newBranchName
      2. 切换分支
         1. git checkout branchName
      3. 创建并切换到创建的分支
         1. git checkout –b newBranchName
      4. 删除分支
         1. git branch –d branchName
      5. 合并分支
         1. git merge branchName
   2. 远程分支-GitHub代替讲解
      1. 新建立一个github仓库
      2. 建立远程分支:git remove add remoeteBranchName url
         1. 【路径: .git/refs/remotes】
      3. 建立远程分支与本地分支的关联
         1. git branch –set-upstream branch\_name
      4. 推送到远程分支git push origin HEAD:master
         1. git push origin 本地提交id:remoteBranchName
         2. git push origin 本地分支:remoteBranchName
         3. git push origin 当前分支id:remoteBranchName
4. 现场
   1. 保存现场
      1. git stash
   2. 查看保存的现场
      1. git stash list
   3. 恢复现场
      1. git stash pop
      2. git stash apply
5. 标签
   1. 创建标签
      1. git tag v1.0.1
   2. 创建带附注的标签
      1. git tag –a v1.0.1 –m
   3. 查看tag详细信息
      1. git show tag\_name
   4. 查看tag列表
      1. git tag
      2. 列出符合条件的标签: git tag –l v\*
   5. 删除标签
      1. git tag –d tag\_name
   6. 基于已有的commit创建tag
      1. git tag tag\_name commit\_id
   7. 推送tag到server
      1. 推送特定的标签:git push origin tag\_name
      2. 推送所有的标签:git push orgin -- tags
   8. 总结:tag 实际上指向的是一个commitId，打标签是为了我们后期需要某个版本是方便我们查阅。
6. 使用Git遇到的问题以及得出的套路
   1. 遇到的问题
      1. 场景1：
         1. 主干有紧急Bug，但是此时我们的开发是在主干上进行的。
      2. 场景2：
         1. 主干有紧急功能，但是此时我们的开发是在主干上进行的。
   2. 如何解决上述问题:
      1. 在没使用分支的情况下
         1. git checkout commitId
         2. git branch –b newBranchName
         3. git push
      2. 在分支使用合理的情况下，上述问题一版不会存在
      3. 所以总结如下
         1. 在开发速度快，上线速度慢的情况下，每个成模块的功能要使用一个单独的分支。例如上面的情况或者有A,B,C三个模块需要开发，你开发完了A模块，此时QA测试或者等上线流程。此时你可以新建立一个分支继续开发B模块，以及C模块，A上线完毕之后，将B以及C merge回主干即可。
         2. 每个bug都要使用一个分支,分支名字建议用icafe的bug单号例如[git checkout –b fixbug-1234]，解决完bug再删除分支即可。