Git常用命令

git clone ’url‘; 克隆项目到本地

git fetch 获取最新分支

git pull 获取最新提交

git checkout -b ‘branchName’ 切个人分支

git add . 添加所有更改到暂存状态

git commit -m ‘提交备注信息’ 提交代码到本地

git push 推送代码到远端

git stash 将当前更改保存

git stash pop 检出保存的文件

git status 查看当前提价状态

git log 查看提交日志

git revert 回滚到上一个版本（新代码将会永久删除） 慎用

Git 操作流程

从拉取项目到提交代码到远端的基本流程：

git clone ’url‘; 克隆项目到本地

git pull 获取最新提交

git fetch 获取最新分支

git checkout -b ‘yourself branchName’ 从当前分支检出个人分支

git pull => git push 将本地代码代码推送到远程 保持远端和本地一致

。。。。。。。。。。。开发过程

git add . 将修改文件暂存

git commit -m ‘提交备注信息’ 提交代码到本地

git push 推送代码到远端

以上操作属于比较完整的一个前端开发流程

分支管理

项目分支：

release分支（如果有灰度测试或者uat环境有必要，一般情况下没必要）

master 分支： 整个项目主分支（受保护的，不能更改的）

test分支：测试环境分支

dev分支：开发分支

基于dev的迭代分支（看情况是否要用）

使用场景：多个迭代并行，但是又不同时上线的情况。可以实现单个迭代独立上线。

dev=> feature-weiguohua-20200623 基于开发分支的个人分支 如果存在迭代分支，则为基于迭代分支的个人分支

分支检出顺序：

master分支 => test分支 => dev分支=> 迭代分支 => 个人分支

分支合并顺序：

个人分支 => 迭代分支=> dev分支 => test分支 => master分支

生产环境发布分支一般为master分支

**个人分支维护**

每一期项目对应一个个人分支，并删除上个版本的个人分支。

检出新的分支前先git pull一下dev分支 保证代码最新。

如果需要公共的内容，个人分支上面又没有，可以把dev分支的代码合并到个人分支

开发完成之后把个人分支往dev分支上合并，并解决冲突。

如果在测试过程当中有BUG也在个人分支上修改，合并顺序：个人分支=>dev=test，虽然繁琐但是能保证每个分支上代码一致，冲突也会减少。

**dev分支**

dev分支的代码永远是最新的，迭代开发完成后，每个人的代码都要合并到dev, 开发自测功能。

全部代码都是基于dev分支，不能单独修改dev分支代码，如有特殊情况建议检出一个fix分支，开发完成后合并到dev。

**Test分支**

只作为测试环境发布代码用，禁止修改分支上的代码。每次迭代开发完成，将dev分支的代码合并到test发布到测试环境。

**master分支**

测试完成后将test分支的代码合并到master分支，该分支主要功能是作为发布生产环境。除了管理员外，开发人员将没有任何操作master分支的权限。