1. 默认主干分支为master
2. 在本地的master上新建一个分支(dev)并同时切换到那个分支上：git checkout -b dev master；【等同于：git branch dev（新建分支）+git checkout dev (切换分支)】
3. 编辑本分支的信息，
4. 将分支(dev)push到git上并同时建立远程分支并同时建立远程分支关联：git push --set-upstream origin dev。

【远程分支关联：是如果在本地分支下操作： git pull, git push ，不需要指定在命令行指定远程的分支。】．

若无远程分支关联则是如下情况:

直接将本地分支(dev)push到git上：git push origin dev

直接将网络分支(dev)pull到本地当前分支：git pull origin dev

1. 以后只需要切换到对应的分支(git checkout dev)，做出相应的修改。可以直接输入git pull 或者git push来实现同一分支下的代码管理。
2. [如果本地目前只有master分支。使用git branch -a查看所有的远程分支。 git checkout -b dev origin/dev----checkout远程的dev分支，在本地起名为dev分支，并切换到本地的dev分支。]
3. 先切换到master主分支上(git checkout master)，再合并分支dev(git merge –-no-ff dev)。

【若合并分支时出现冲突(比如除了master，还有两个分支，这两个分支对同一个文件进行了修改)，则git会给予提示，使用git status可以查看冲突的文件，并手动修改好，再push】

1. 删除本地分支：git branch -d dev

删除网络分支:

git branch -r -d origin/branch-name

git push origin:dev

常用指令：

Git checkout XXX ,切换到XXX分支

Git merge XXX 合并分支XXX

提示：

1. 切换到对应的分支时(git checkout dev)，文件夹内的代码(包括数量和内容)，都是会改变的！

背景

执行Git merge 时，默认情况下，Git执行"快进式合并"（fast-farward merge）；会直接将Master分支指向Dev分支。

执行Git merge –no-ff dev时;使用--no-ff参数后，会执行正常合并，在Master分支上生成一个新节点。为了保证版本演进的清晰，推荐采用这种做法。

分支管理

命名规范：

主分支：master

开发时的分支：develop(develop的缩写)

临时分支(临时分支将在develop分支上建立)：

临时功能分支：feature-\*

临时预发布分支：release-1.2

临时修补BUG分支：fixbug-0.1

资料参考

<http://www.ruanyifeng.com/blog/2012/07/git.html>

这个只是个人的笔记，如果有错误或者确实的地方欢迎大家指正~

--Mr.Gao