目的：

    1，大家共同维护一个清晰可读代码提交历史，便于代码审查、代码回滚、剔除事故代码

    2，尽可能减少事故发生

规范一：

    1，每一个新项目从主干（master）开一个新分支

    2，不允许多个项目共用一个分支，哪怕可以同时上线

    3，建议分支名：feature/项目名称，例如：feature/v79

规范二：每次提交代码都必须有含义明确的注释，比如：“修改了XXX的BUG”，“完成XXX功能”，禁止：“调试”，“测试”，“debug”

规范三：推荐频繁的提交代码，但是多人共用一个项目分支时，不要把能导致测试环境崩溃的代码PUSH到远程。

规范四：切换分支之前一定要确保本地工作区是干净的，所有修改和添加的文件都提交后在切换分支

规范五：推荐频繁的将主干分支中的内容合并到项目分支中

规范六：谨慎使用 rebase，reset，revert等改变Git提交历史的操作

规范七：提交点（CommitId）和标签（Tag）不是真正的分支，检出时要加 -b 选项生成新分支，例如：git checkout -b branch\_name tag\_name

规范八：测试及调试时将项目分支合并到测试分支，例如：develop

规范九：每次上线项目分支合并到主干，马上打一个轻量级的Tag，例如：git tag v20180509.1

规范十：禁止直接在主干分支上进行修改、提交、推送等操作，如果需要紧急修改线上BUG应新开热修复分支，例如：hotfix/20180509-1

场景一：创建新功能分支

    #检查本地工作区是否干净，如果有修改，添加并提交新建和修改的文件

    git status

    git add filename

    git commit -a -m '注释'

    #切换到主干，并拉取最新的代码

    git checkout master

    git fetch -p

    git pull --ff-only

    #从主干拉出项目分支

    git checkout -b feature/v79

场景二：日常提交代码及分支合并

    #切换到项目分支并拉取最新内容

    git checkout feature/v79

    git fetch -p

    git pull --ff-only

    #添加新建文件到缓冲区

    git add filename

    #自动添加修改文件到缓冲区并提交

    git commit -a -m '注释'

    #推送分支代码到远程仓库

git push origin feature/v79/mh

#测试合并到测试分支

git checkout develop

    git fetch -p

    git pull --ff-only

git merge --no-ff feature/v79

git push origin develop

场景三：上线及打Tag

    #切换到主干及合并代码

    git checkout master

    git fetch -p

git pull --ff-only

git merge --no-ff feature/v79

git push origin master

    #打Tag

git tag v20180509.1

git push origin v20180509.1