**提交规范**

Git提交注释commit规范：

Add：新功能

Upd：新功能的完善，补充

Fix：修补bug号，缺陷，优化

Refactor：重构（即不是新增功能，也不是修改bug的代码变动）

例子：

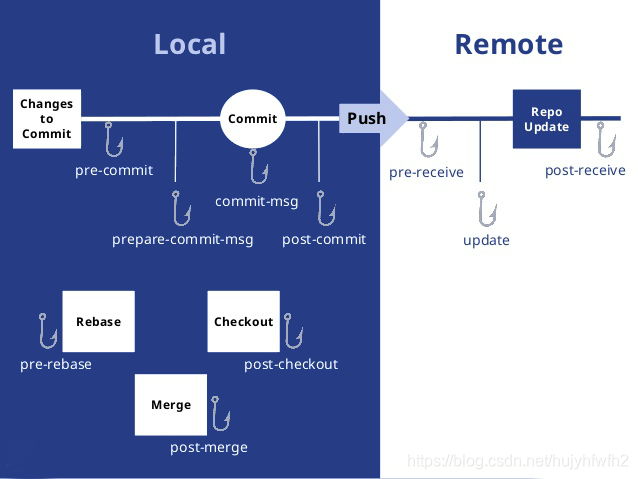
Add：xxx功能，具体某个新增、修复功能点描述信息

Upd：xxx功能，具体某个完善、补充功能点描述信息

Fix：#123，修复bug的描述信息

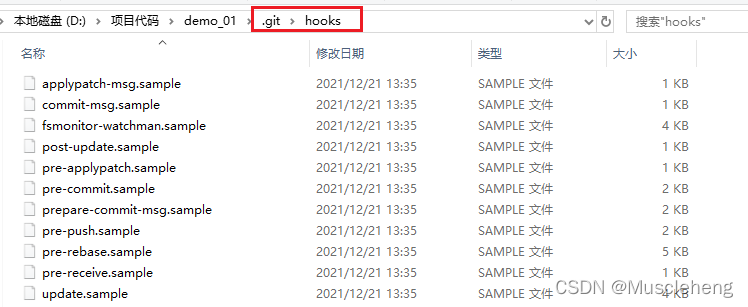
以上规范在git服务器端进行push之后，。强制验证合规性，如果不合规，git会显示例子。

介绍验证方式Git Hooks：

Git hooks上分为两类：客户端钩子和服务器端钩子,如下 

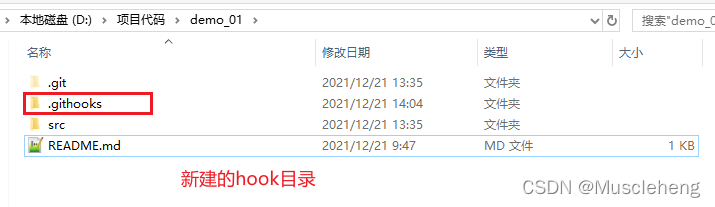
## ****1、 客户端钩子****

项目的hooks如下,地址为当前项目的.git/hooks目录下，打开后可以看到如下文件，里面有示例shell脚本，默认是不启用的，只有去掉.sample后缀才能生效，里面的脚本代码可以根据实际情况编写。



默认客户端hook是在项目的.git/hooks/ 目录下，默认是独立项目之外，不受版本控制的，如果要修改脚本，需要每个开发都复制一份或者都自己修改，比较麻烦。因此可以通过修改客户端配置来实现hooks受git版本控制。

如果需要使其随项目受控制，需要自定义一个目录存放hook脚本，例如：在项目.git目录同级目录下新建.githooks目录：



然后执行命令把默认目录改为我们的自定义目录，在.githooks同级目录下执行git命令： git config --global core.hooksPath .githooks

Email要设置成公司邮箱：

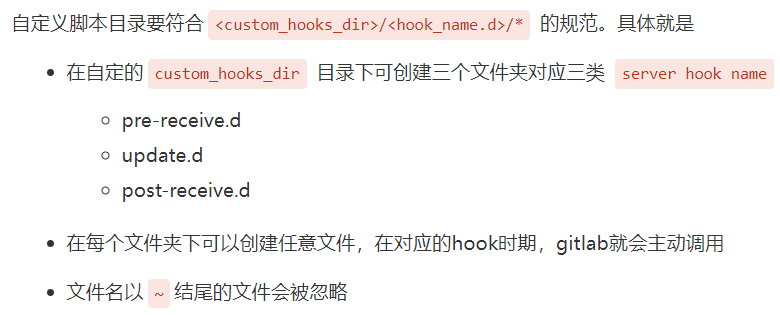
此时就可以把 .githooks包含到项目里跟随项目了

2、 服务端钩子

gitlab配置：

gitaly['**custom\_hooks\_dir**'] = "/home/gitlab/hooks"

创建hook文件：





在 pre-receive（推送前）阶段来做提交信息的校验，如果不符合我们的要求，直接返回，则该推送便不会推送到 GitLab 仓库中去。

