## 主要概念

### 版本管理

无论是传统的SVN，还是当前的Git，这一类版本管理工具主要解决软件开发过程中两大问题：

* **历史版本管理**：软件开发过程中，有时需要回退到某个历史时刻点的状态版本。这种情景下，版本管工具就是“时间机器”。
* **多人协作管理**：庞大的软件项目往往需要拆分成独立而又相互关联的不同模块，交由不同的人员开发、维护。因此，引入“分支”概念，可以保证不同模块的开发人员可以在不同的分支上进行开发，并且相互不影响。待时机成熟，各模块分支又可以合并成一个整体，好比，树枝各自生长，最终又合并到主干。

### 远程仓库与本地仓库（repository）

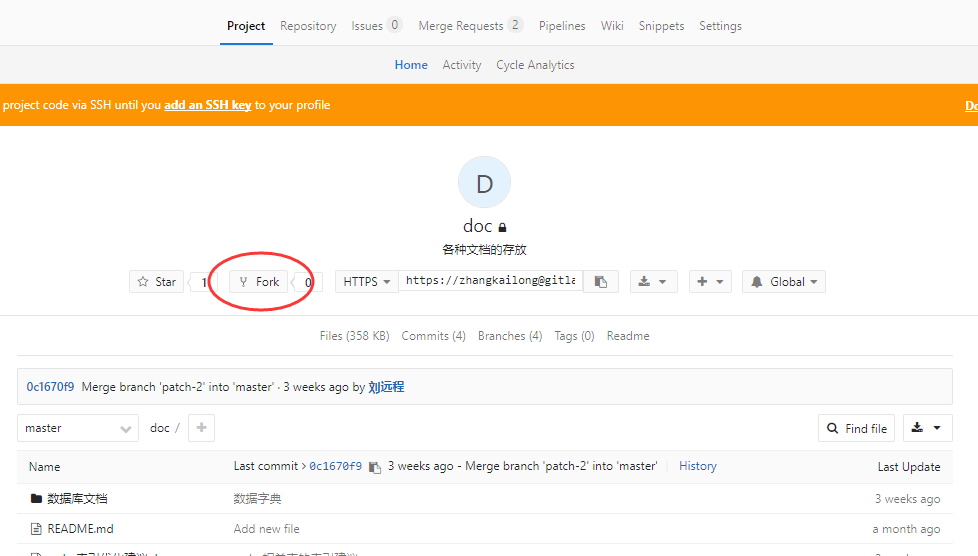
开发人员安装完Git客户端后，便可以在本地计算机创建本地仓库，与此对应，Git服务端存放的便是远程仓库。一般情况下，开发人员初始化配置本地仓库之后，便可以对本地仓库中项目直接进行开发，然后将相关修改内容提交至本地仓库。如果条件适当，开发人员便将修改后的本地仓库项目推动到远程仓库——团队中其他开发人员便能查看、应用相关修改内容。

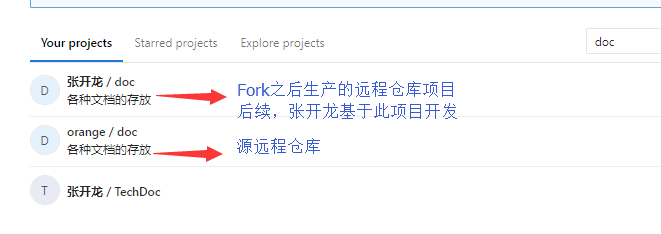
## 使用教程

使用Git，可以直接使用命令行模式，也可在IDE图形化界面直接使用。考虑到不同的IDE界面有所不同，下文主要介绍下命令行方式。通**过git进行协同开发有多种模式，如果侧重于“merge request”的方式，读者可以略读步骤3、步骤5。**

### Fork远程仓库：

浏览器控制台找到对应远程仓库项目，执行Fork操作之后，自动生成当前最新状态的项目副本，如下图：





### 克隆远程仓库到本地仓库:

安装完git客户端之后，在本地任意目录下，通过下述命令可以将远程仓库中最新状态的代码同步到本地。其中，克隆地址可以通过浏览器直接复制，如下图A。克隆成功后，本地计算机自动生成对应项目仓库目录，如下图B。

$ git clone https://zhangkailong@gitlab.lvqu.me/zhangkailong/doc.git

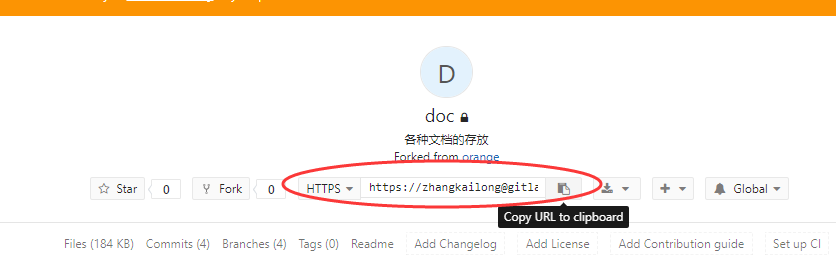
Cloning into 'doc'...

remote: Counting objects: 22, done.

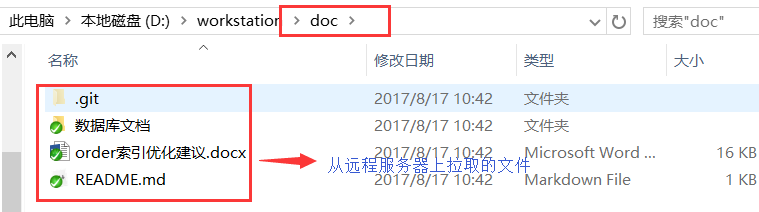
remote: Compressing objects: 100% (16/16), done.

remote: Total 22 (delta 2), reused 22 (delta 2)

Unpacking objects: 100% (22/22), done.



图A



图B

### 分支创建与切换：

远程仓库项目克隆到本地仓库之后，开发人员可以在本地仓库选择不同的分支进行开发、调试（本教程选择master分支进行演示）。分支管理的基本操作如下：

//在本地仓库创建新的分支

$ git branch new\_Branch\_Name

//删除某一个分支

$ git –d Branch\_Name

//切换至分支“new\_Branch\_Name”

$ git checkout new\_Branch\_Name

//查看所有本地仓库的分支名，当前所在的分支，以星号与高亮文本区分体现

$ git branch --list

\* develop\_v3

master\_v3

### 分支基本操作：Commit、push、pull、fetch

开发人员修改本地项目的develop\_v3分支后，需要通过提交（commit）操作之后，将修改保存至本地仓库的develop\_v3分支；通过push操作在将本地仓库的当前对应分支推送、更新到远程仓库。

与此同时，如何团队中其他成员也更新了远程仓库的develop\_v3分支，你也可以通过pull或fetch操作将远程仓库的分支同步到本地仓库的develop\_v3分支。

$ git commit -–all

$ git push remote-resposity remote-branch

$ git pull

### 合并分支：

开发人员如果想将分支A上修改应用到分支B上，则可以通过git merge实现。以上述步骤3为例：本地仓库项目中已有develop\_v3和master\_v3两个分支，你在本地develop\_v3分支上进行开发。这时，项目主管告知你“远程仓库中的master分支已经更新修正若干bug”，你当前的开发工作也要基于最新版本的展开。——如何将远程仓库中的master分支的修改应用到本地仓库的develop\_V3f分支？

//切换到本地master分支

$ git checkout master

//将远程仓库中master分支拉取到本地仓库的master分支

$ git pull

//切换到本地develop\_v3分支

$ git checkout develop\_v3

//将本地仓库的master分支合并到develop-v3分支:将master分支的修改应用到develop\_v3

$ git merge master

### 添加远程仓库

通过上述步骤，我们基本实现了远程仓库与本地仓库之间的相互同步与分支切换。但是，存在一个问题：如果被fork的源远程仓库“orage/doc”更新后，本地仓库与对应的“张开龙/doc”远程仓库，如何更新？

**首先，**通过$ git remote可以查看本地仓库对应的远程仓库的地址

$ git remote -v

origin https://zhangkailong@gitlab.lvqu.me/zhangkailong/doc.git (fetch)

origin https://zhangkailong@gitlab.lvqu.me/zhangkailong/doc.git (push)

**然后，**添加远程仓库：将“源”远程仓库也添加为本地仓库的远程仓库。一般的，我们也将“源”远程仓库称之为“**上游远程仓库”**，因为我们直接远程仓库从该仓库fork而来。

//为本地仓库添加远程仓库，支持自定义远程仓库名为“orage”

$ git remote add orage <https://zhangkailong@gitlab.lvqu.me/orange/doc.git>

//查看远程仓库，发现已经有两个远程仓库

$ git remote -v

orage https://zhangkailong@gitlab.lvqu.me/orange/doc.git (fetch)

orage https://zhangkailong@gitlab.lvqu.me/orange/doc.git (push)

origin https://zhangkailong@gitlab.lvqu.me/zhangkailong/doc.git (fetch)

origin https://zhangkailong@gitlab.lvqu.me/zhangkailong/doc.git (push)

最后，我们可以在本地仓库中通过git pull指令，将上游远程仓库（orage）中的数据同步过来。而后，可以进行直接修改或者push到origin远程仓库。

//基本语法git pull remote-resposity-name branch-name

$ git pull orage master

From https://gitlab.lvqu.me/orange/doc

\* branch master -> FETCH\_HEAD

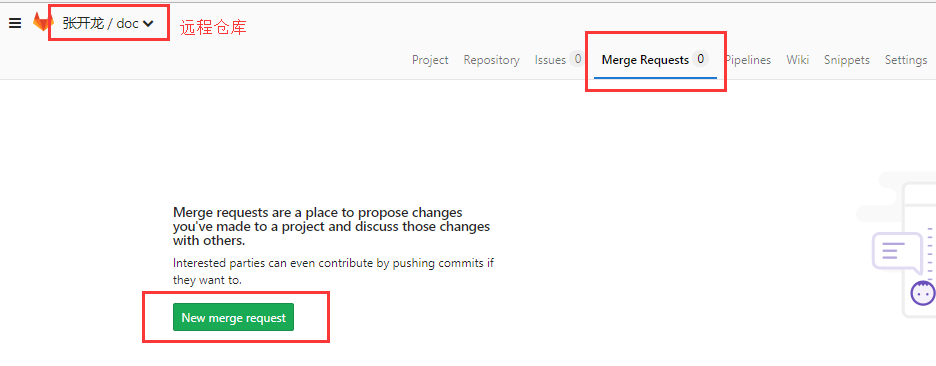
\* [new branch] master -> orage/master

Already up-to-date.

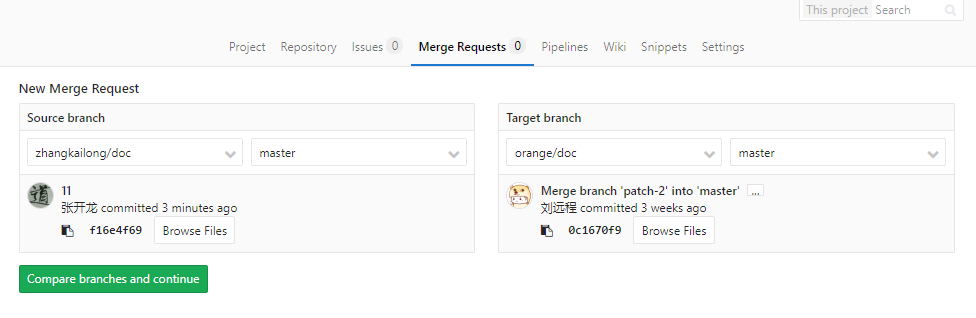
### Merge request

首先，基于步骤6之后，开发人员在更新后的本地仓库分支上进行修改后，通过git push将本地仓库同步至远程仓库origin。

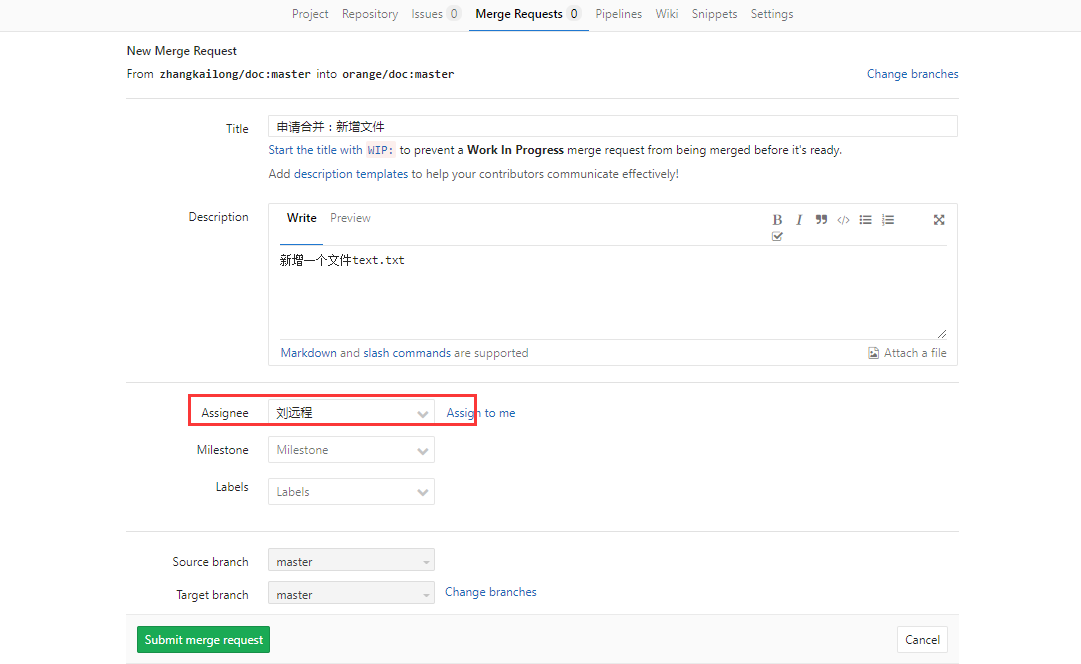
然后，我们通过浏览器登陆“张开龙/doc”（即origin）远程仓库，通过merge request提交合并申请：将origin远程仓库中的修改合并到上游远程仓库“orage/doc”。



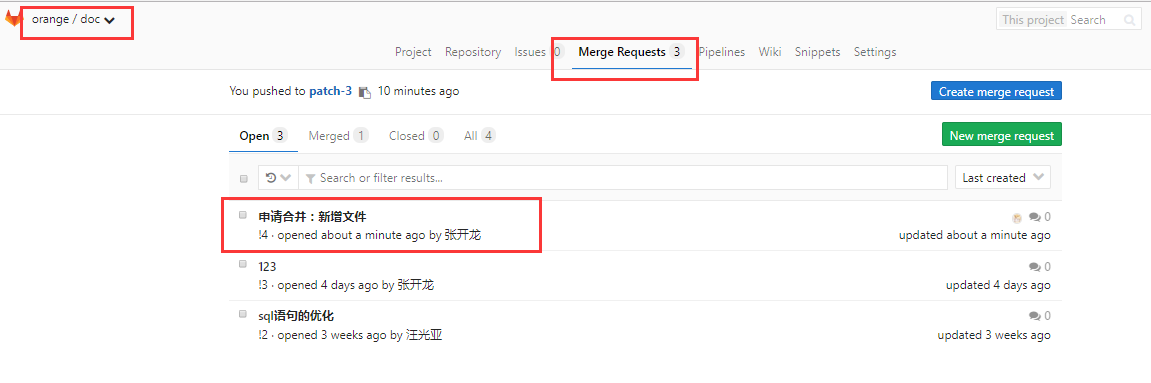
发起合并申请



指定合并的远程仓库的名称与分支名



合并说明与指定合并审核者



提交合并后，上游远程仓库的管理员便可以查看到最新的合并请求