## Gitlab项目流程详细版

**一、基本概念**

1.仓库（Repository）

①源仓库(线上版本库)

在项目的开始,项目的发起者构建起一个项目的最原始的仓库,称为origin。源仓库的有两个作用：

1.汇总参与该项目的各个开发者的代码

2.存放趋于稳定和可发布的代码

源仓库应该是受保护的，开发者不应该直接对其进行开发工作。只有项目管理者能对其进行较高权限的操作。

②开发者仓库(本地仓库)

任何开发者都不会对源仓库进行直接的操作，源仓库建立以后，每个开发者需要做的事情就是把源仓库的“复制”一份，作为自己日常开发的仓库。这个复制是gitlab上面的fork。

每个开发者所fork的仓库是完全独立的，互不干扰，甚至与源仓库都无关。每个开发者仓库相当于一个源仓库实体的影像，开发者在这个影像中进行编码，提交到自己的仓库中，这样就可以轻易地实现团队成员之间的并行开发工作。而开发工作完成以后，开发者可以向源仓库

发送pull request，请求管理员把自己的代码合并到源仓库中，这样就实现了分布式开发工作和集中式的管理。

2.分支（Branch）

①master branch：主分支

master：主分支从项目一开始便存在，它用于存放经过测试，已经完全稳定代码；在项目开发以后的任何时刻当中，master存放的代码应该是可作为产品供用户使用的代码。所以，应该随时保持master仓库代码的清洁和稳定，确保入库之前是通过完全测试和代码reivew的。master分支是所有分支中最不活跃的，大概每个月或每两个月更新一次，每一次master更新的时候都应该用git打上tag，来说明产品有新版本发布。

②develop branch：开发分支

develop：开发分支，一开始从master分支中分离出来，用于开发者存放基本稳定代码。每个开发者的仓库相当于源仓库的一个镜像，每个开发者自己的仓库上也有master和develop。开发者把功能做好以后，是存放到自己的develop中，当测试完以后，可以向管理者发起一个pull request，请求把自己仓库的develop分支合并到源仓库的develop中。所有开发者开发好的功能会在源仓库的develop分支中进行汇总，当develop中的代码经过不断的测试，已经逐渐趋于稳定了，接近产品目标了。这时候，就可以把develop分支合并到master分支中，发布一个新版本。

注:任何人不应该向master直接进行无意义的合并、提交操作。正常情况下，master只应该接受develop的合并，也就是说，master所有代码更新应该源于合并develop的代码。

③feature branch：功能分支

feature：功能性分支，是用于开发项目的功能的分支，是开发者主要战斗阵地。开发者在本地仓库从develop分支分出功能分支，在该分支上进行功能的开发，开发完成以后再合并到develop分支上，这时候功能性分支已经完成任务，可以删除。功能性分支的命名一般为feature-\*，\*为需要开发的功能的名称。

1. **实例操作**

**准备工作：**

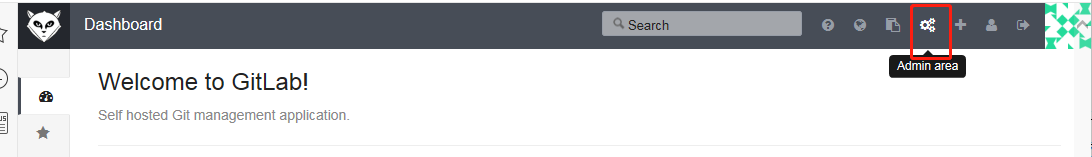
**创建组**

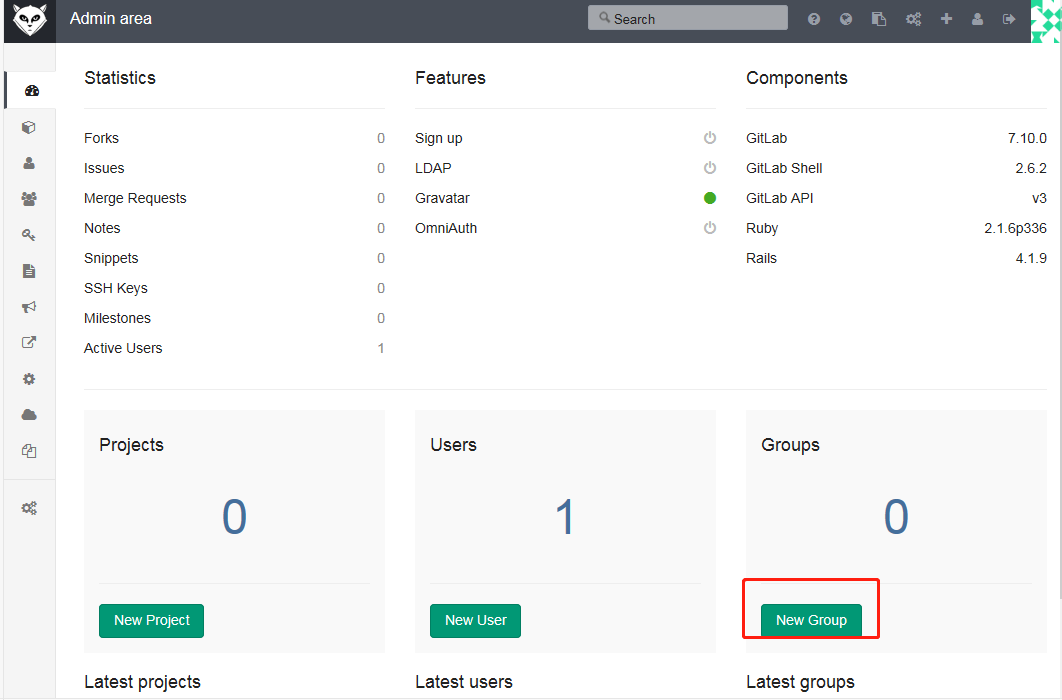
点击gitlab管理区域（Admin area），里面可以创建项目（project），创建用户（user），创建组（group）

说明：project就是开发项目创建，可理解为创建项目仓库，通过项目仓库地址可以实现将项目代码拉取克隆到本地进行开发。

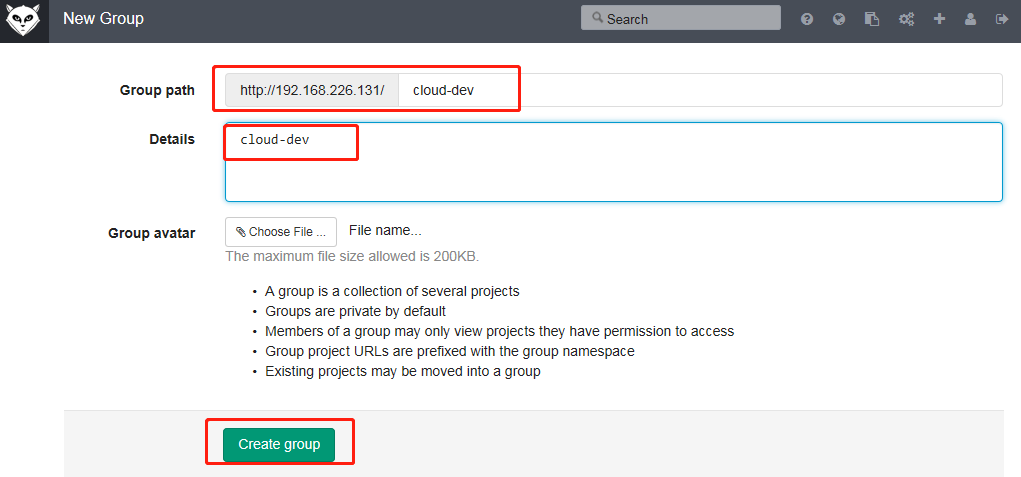
user就是用户，是登陆gitlab的账户，也是克隆代码的账户。

group是组，可以将用户加入到组里。进行关联用户和项目。



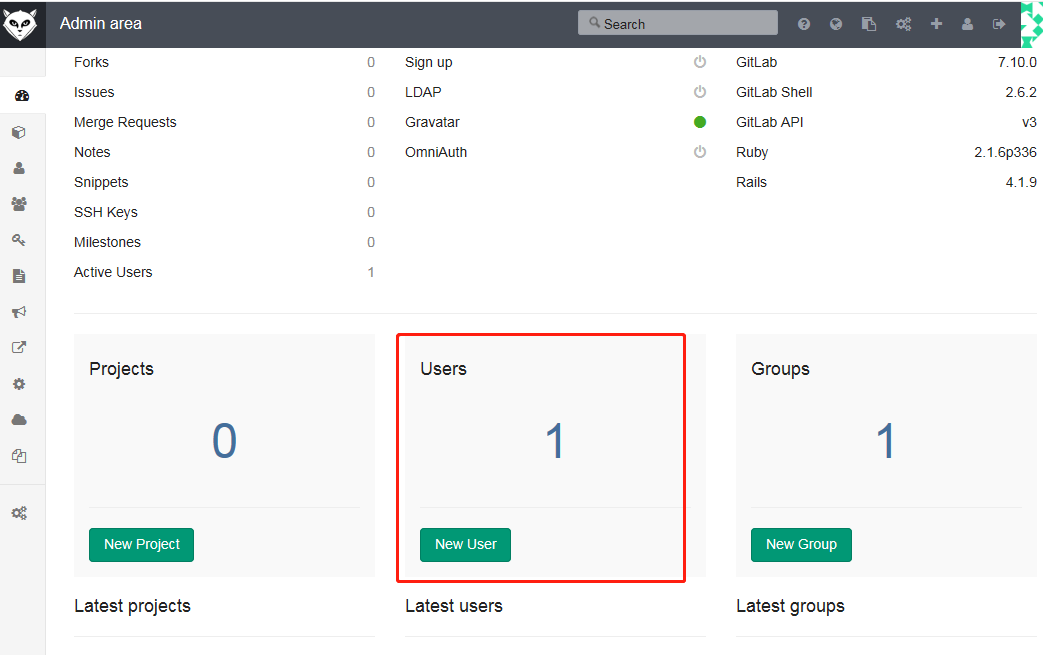


创建一个叫做cloud-dev的组。

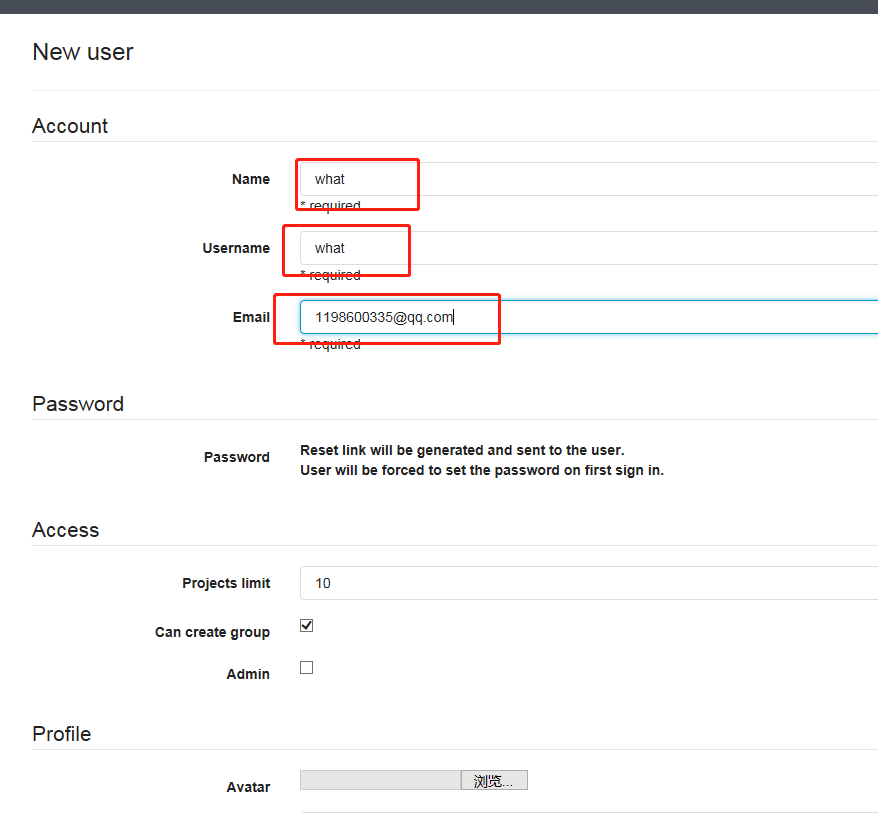


**创建用户**

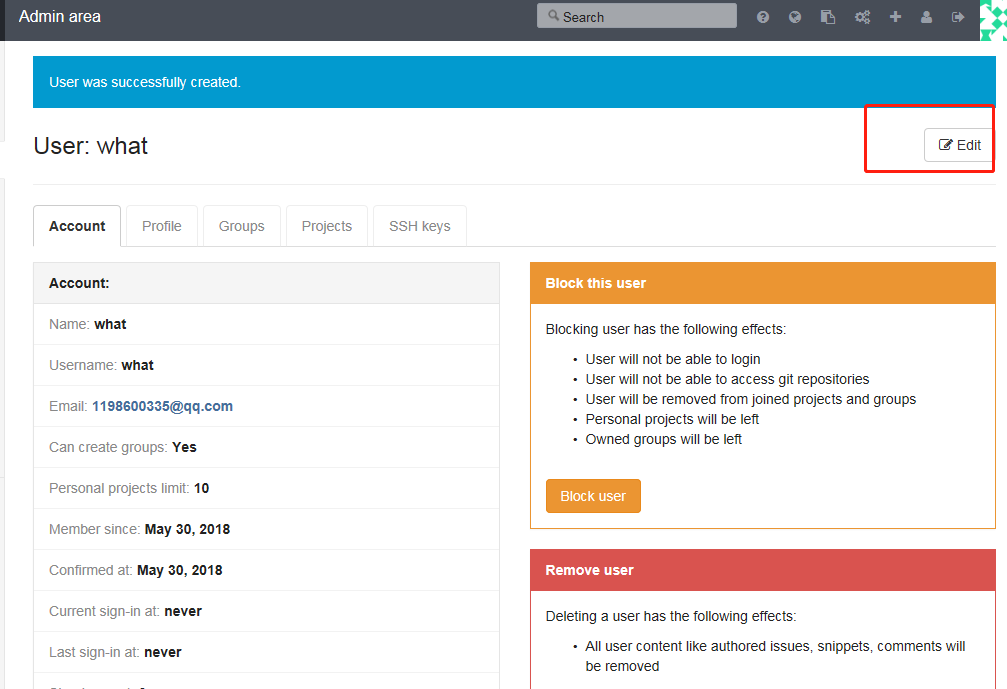
默认的gitlab的管理员用户是Administrator，其中已经有一个用户了，我们在创建一个what用户，作为开发者测试用户。



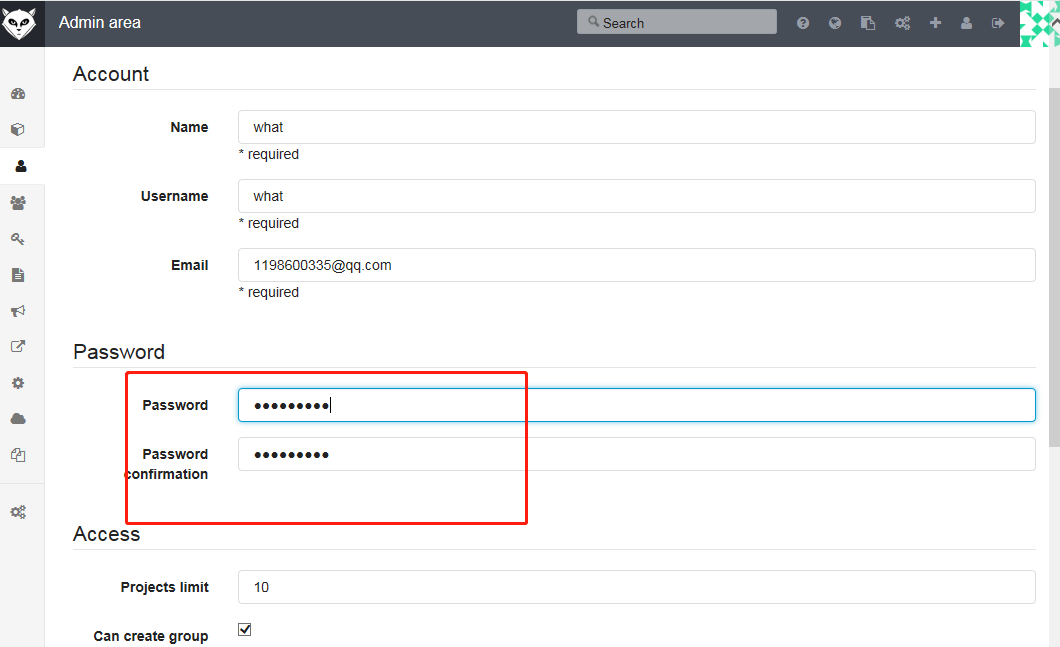
创建用户填，Name就是姓名，Username就是登陆gitlab的用户名，Email是写真实的邮箱账号。此时创建账号，不能填写密码。必须创建账号完成以后，在编辑设置密码。



编辑用户设置密码

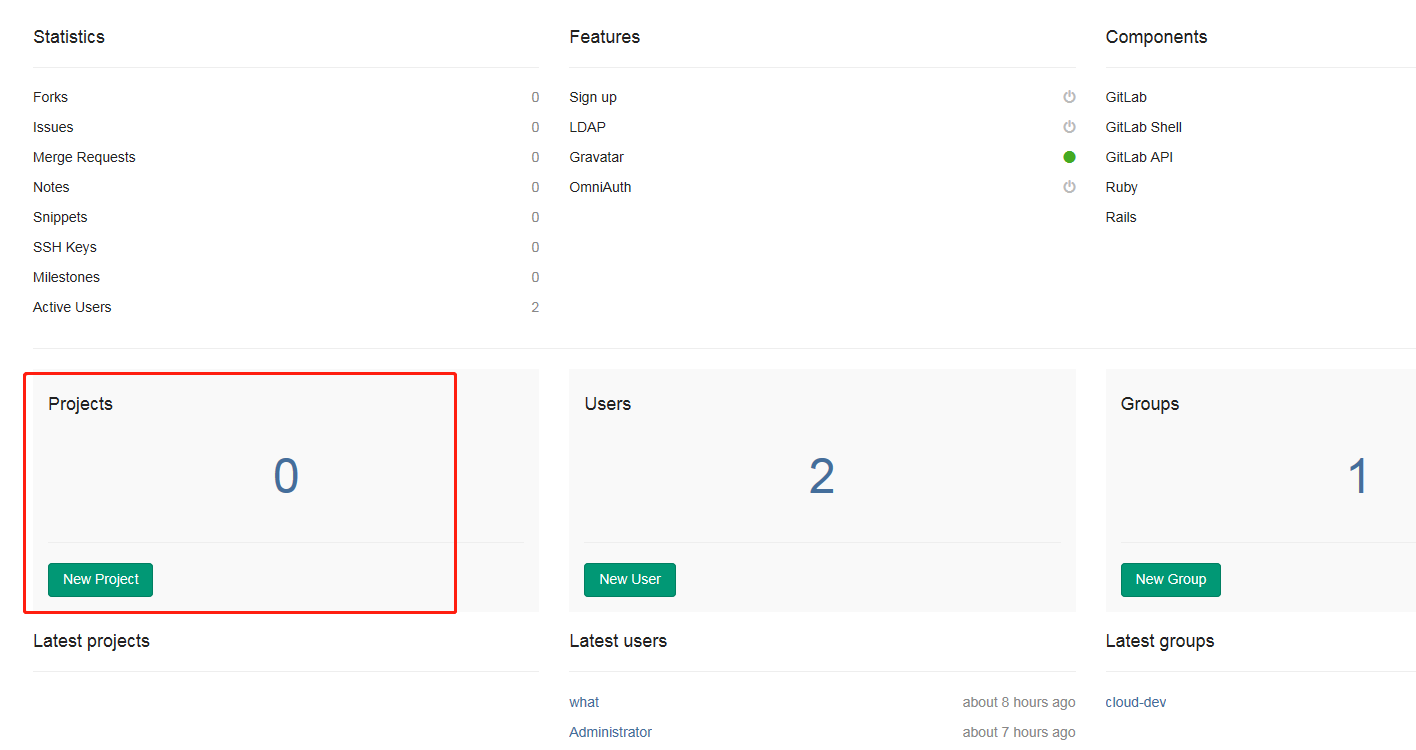


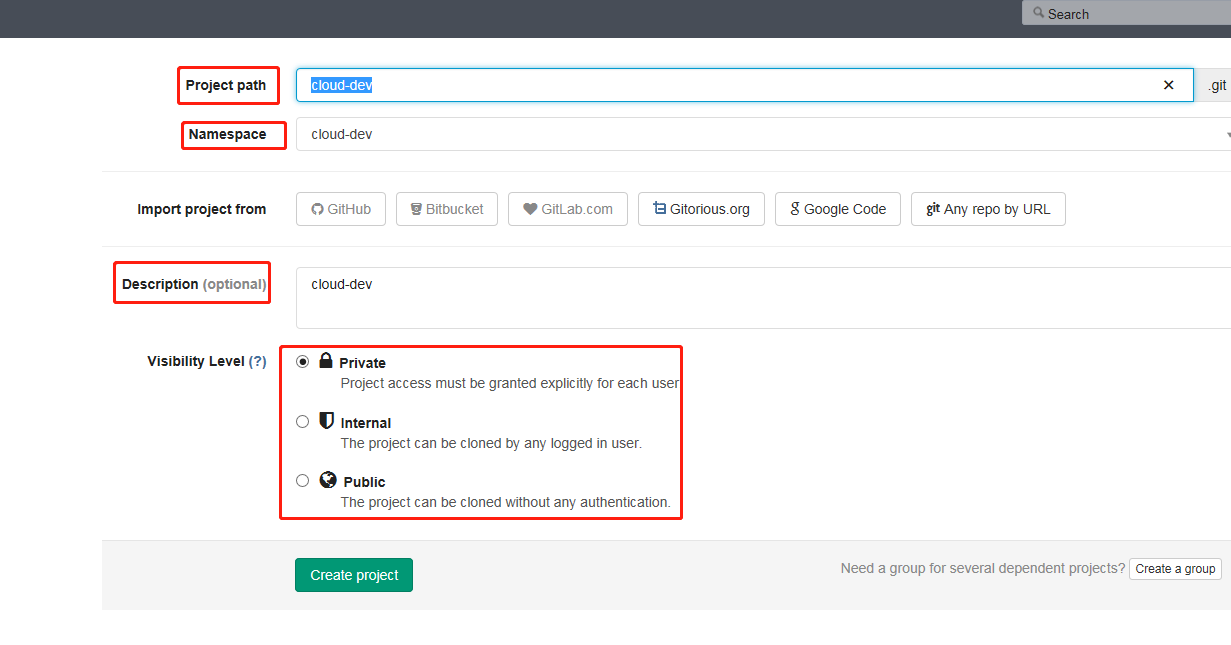
设置好密码以后点保存，save changes



**创建项目**

我们创建一个cloud-dev项目。

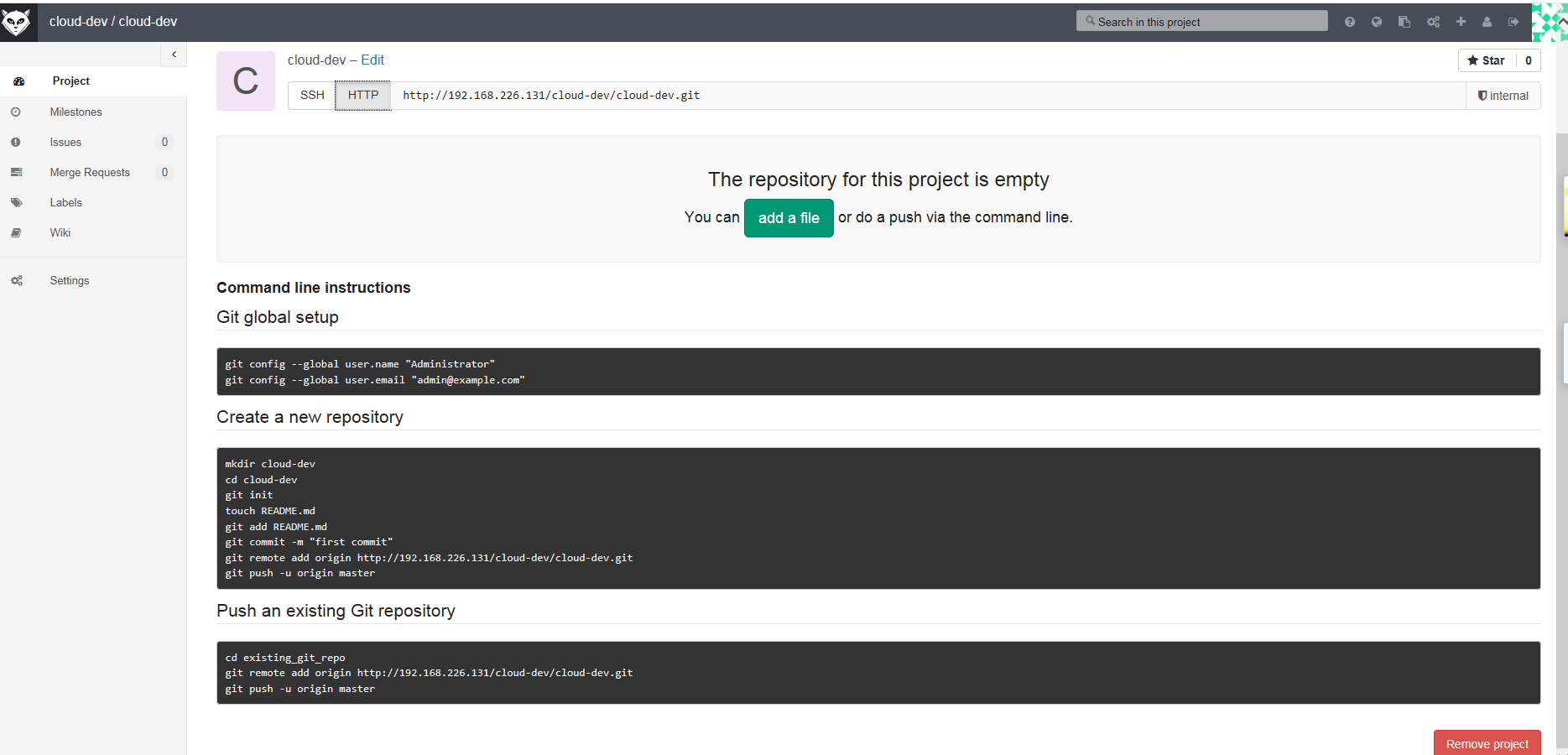




说明：Project path填写项目名称；Namespace可以理解为选择组，这里我们选择刚刚创建的cloud-dev组；import project from这个默认不用管，意思是从其他的地方获取项目；Description这个是写描述信息，项目的简介；visitlity level是权限级别，Private是私有权限，只有此组的用户和管理员有权限查看此项目的代码；Internal是内部级别权限，只要能登陆到此gitlab的所有账号，都可以访问此项目；Public是公共权限，任何人，就算没有登陆也有权限访问此项目。建议选择Private，可以对权限进行精确控制。

创建项目完成以后，自动进入项目界面。

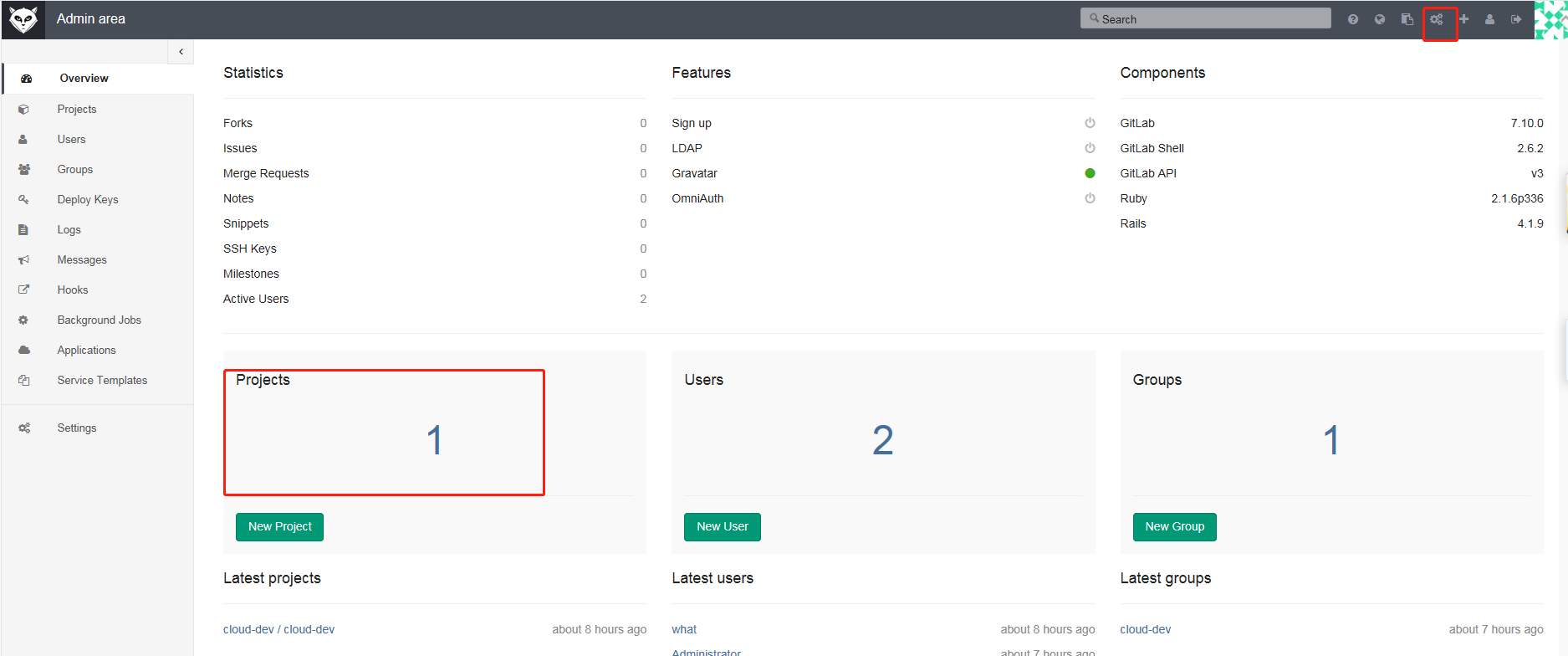
说明：访问项目仓库两种方式，第一种ssh，第二种http，建议用第二种就可以了。



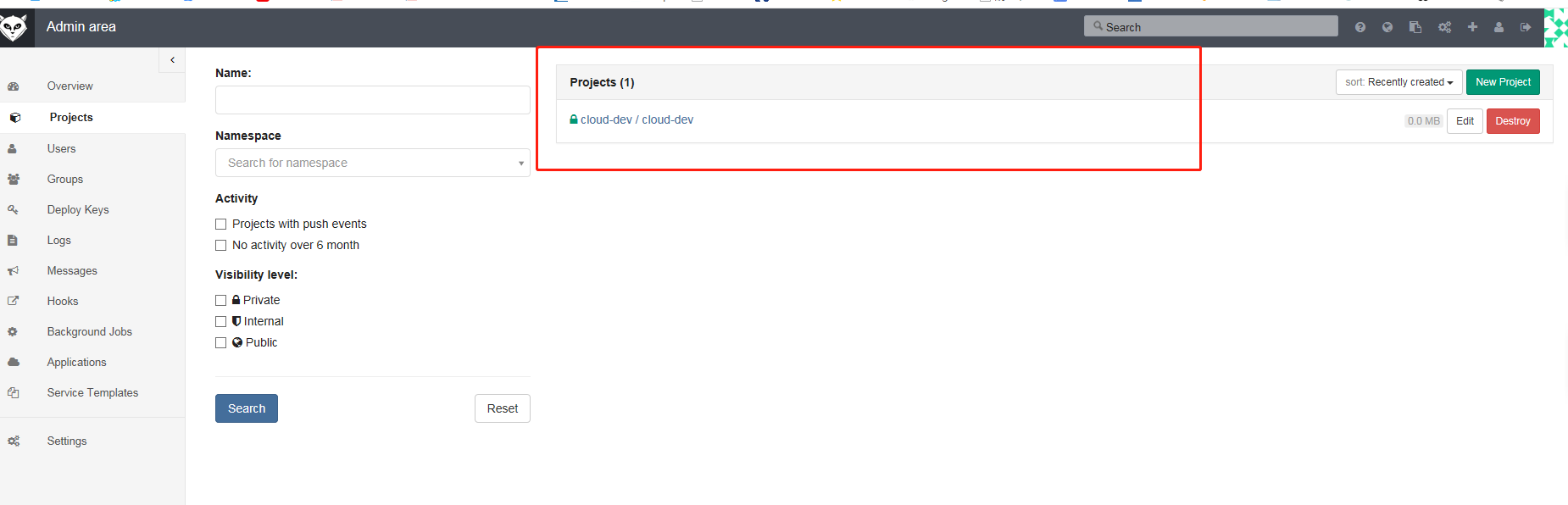
cloud-dev仓库创建好了，需要关联用户了。

详细步骤进入项目的设置

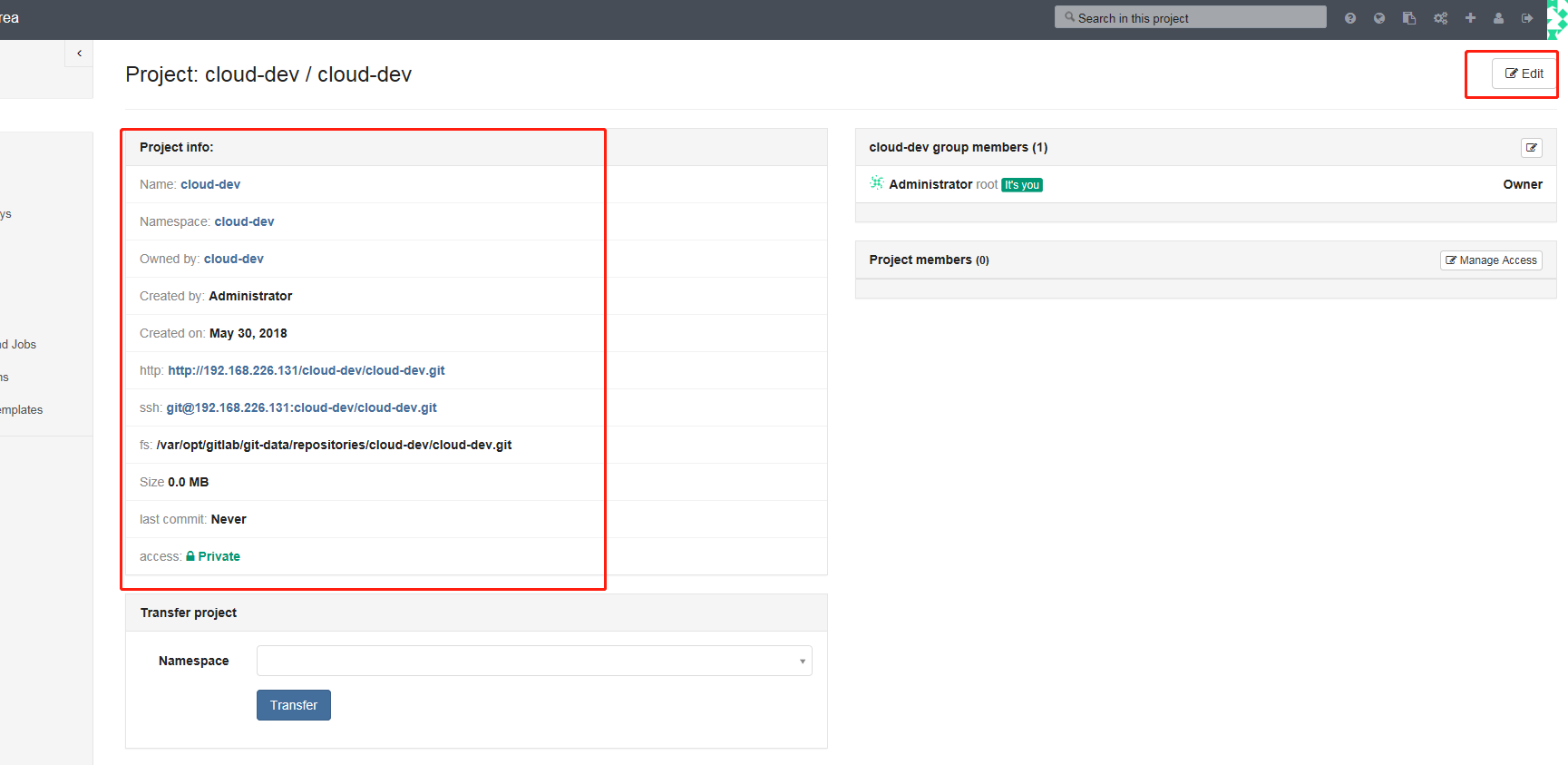
点管理区域（Admin area），选择项目进去。



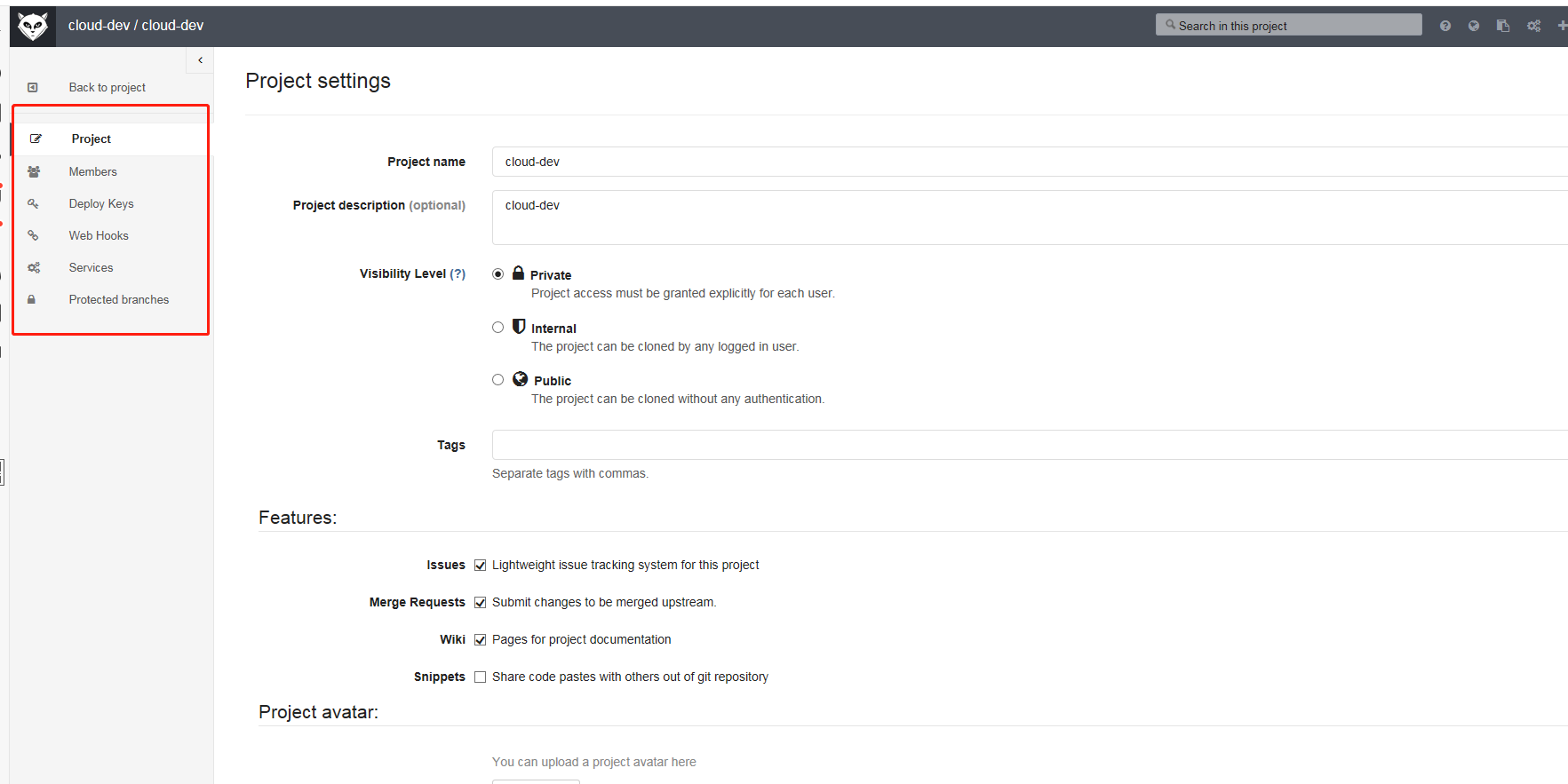
点进去以后是这样，我们选择cloud-dev项目进去。



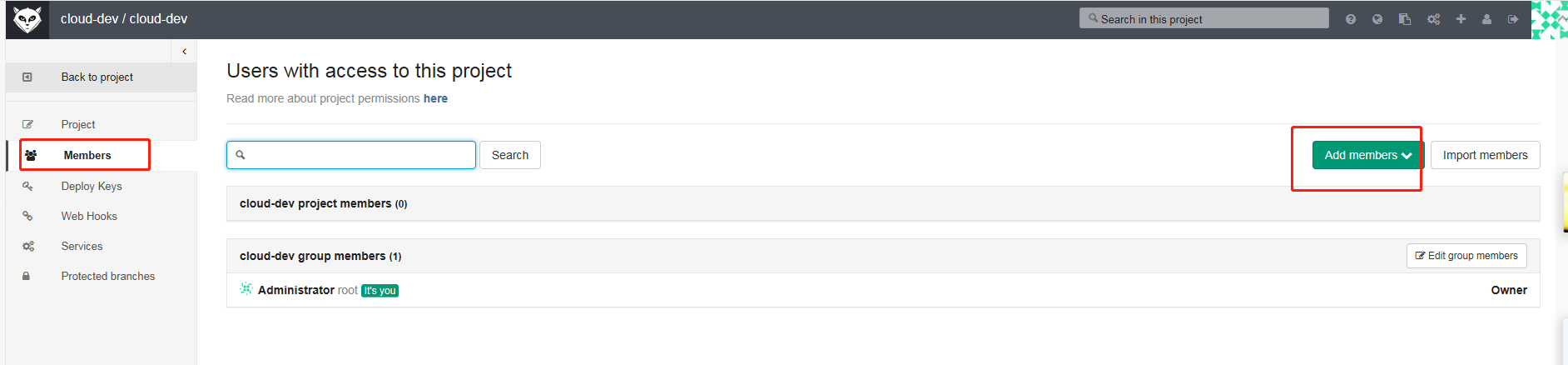
可以看到Project info就是项目的详细信息，包括权限，创建人，项目地址等信息。



我们点右上角的编辑，进入项目的编辑。进行关联用户。项目设置在左边。我们点Members，可以进行关联用户。



点添加用户，选择what用户添加。



设置what用户对项目的权限为开发者（Developer）其他权限解释：

Guest Reporter Developer Master Owner

|  |
| --- |
|  |
| 创建新问题 | √ | √ | √ | √ | √ |
| 留言 | √ | √ | √ | √ | √ |
| 版本列表可见 | √ | √ | √ | √ | √ |
| 生成日志可见 | √ | √ | √ | √ | √ |
| 下载和浏览生成构件 | √ | √ | √ | √ | √ |
| 拉取项目代码 |  | √ | √ | √ | √ |
| 下载项目 |  | √ | √ | √ | √ |
| 创建代码片段 |  | √ | √ | √ | √ |
| 管理问题跟踪器 |  | √ | √ | √ | √ |
| 管理标签 |  | √ | √ | √ | √ |
| 查看提交状态 |  | √ | √ | √ | √ |
| 查看容器注册表 |  | √ | √ | √ | √ |
| 管理合并请求 |  |  | √ | √ | √ |
| 创建新的合并请求 |  |  | √ | √ | √ |
| 创建新的分支 |  |  | √ | √ | √ |
| 推送到不受保护的分支 |  |  | √ | √ | √ |
| 强制推送到不受保护的分支 |  |  | √ | √ | √ |
| 删除非保护分支 |  |  | √ | √ | √ |
| 添加标签 |  |  | √ | √ | √ |
| 写一个wiki |  |  | √ | √ | √ |
| 取消和重试生成 |  |  | √ | √ | √ |
| 创建或更新提交状态 |  |  | √ | √ | √ |
| 更新容器注册表 |  |  | √ | √ | √ |
| 删除容器注册表图像 |  |  | √ | √ | √ |
| 创造新的里程碑 |  |  |  | √ | √ |
| 添加新的团队成员 |  |  |  | √ | √ |
| 推送到保护的分支 |  |  |  | √ | √ |
| 启用/禁用 分支保护 |  |  |  | √ | √ |
| 修改/删除git标签 |  |  |  | √ | √ |
| 添加部署项目的密钥 |  |  |  | √ | √ |
| 配置项目挂钩 |  |  |  | √ | √ |
| 管理runners |  |  |  | √ | √ |
| 管理建立触发器 |  |  |  | √ | √ |
| 管理变量 |  |  |  | √ | √ |
| 变更可视级别 |  |  |  |  | √ |
| 将项目转移到另一个命名空间 |  |  |  |  | √ |
| 删除项目 |  |  |  |  | √ |
| 强制推送到保护分支 |  |  |  |  |  |
| 移除受保护分支 |  |  |  |  |  |
|  |  |  |  |  |  |

权限解释：

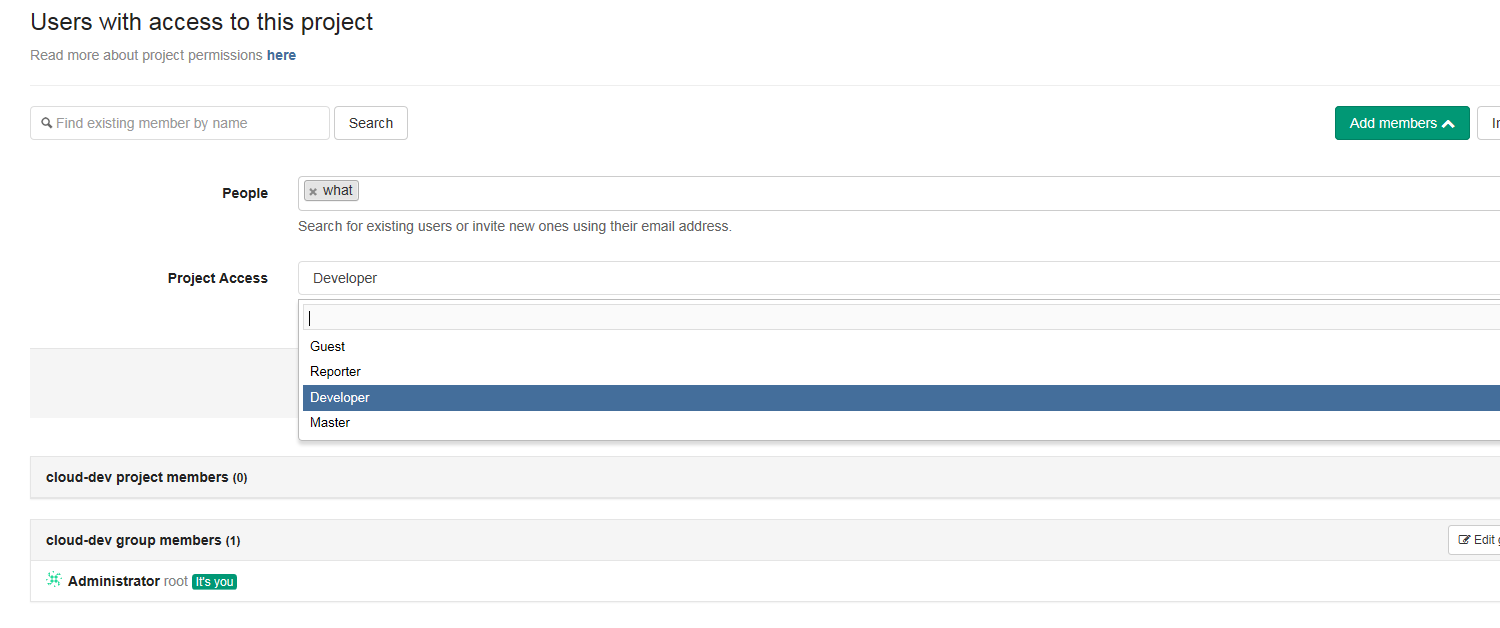
Guest：访客，貌似没有什么权限；

****Reporter****：可以使用 项目地址拉代码，但是不能push 到仓库的默认分支；

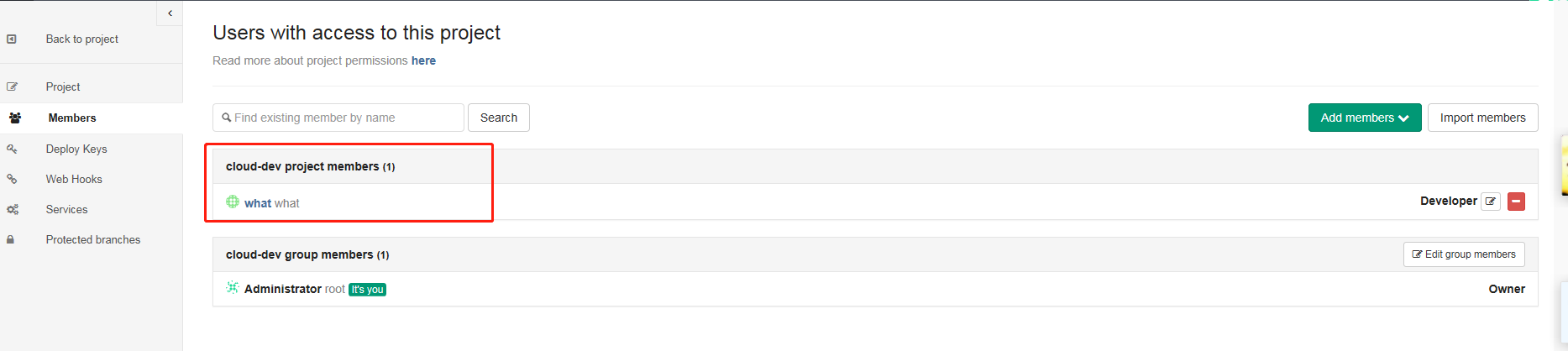
****Developer****：项目的开发人员，能够推送和删除没有保护的分支，刚创建的分支 默认 都是没有保护的；

****Master****：项目管理人员，可以对没有保护和有保护的所有分支进行操作，几乎拥有所有权限；

Owner：系统管理员，拥有所有权限；



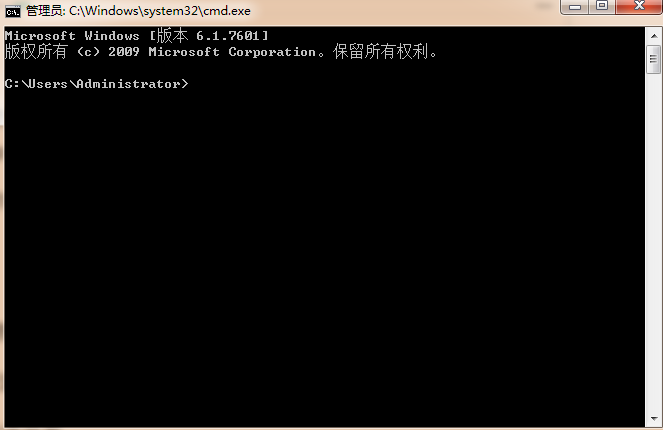
创建完成有提示，说用户加入了该项目中。



1. 构建源仓库

上述步骤已经创建了一个cloud-dev的项目了。下面将进行初始化仓库的操作以及master和develop分支的创建。

Win7本地git操作初始化仓库，win7系统需要安装git客户端以后才能进行如下操作。打开cmd命令行。进行命令操作。



设置用户

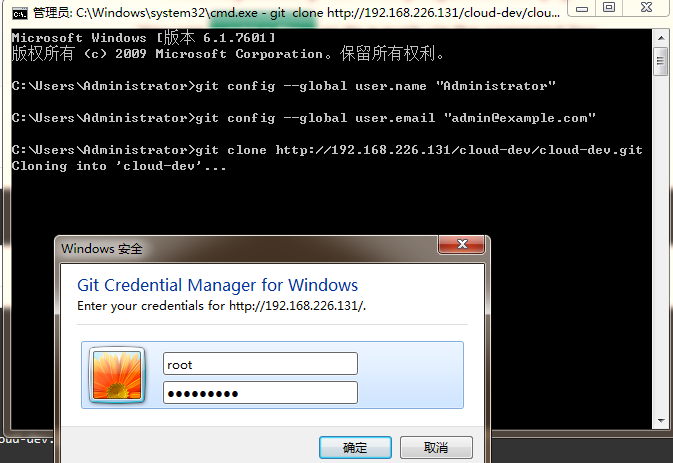
git config --global user.name "Administrator"

设置email

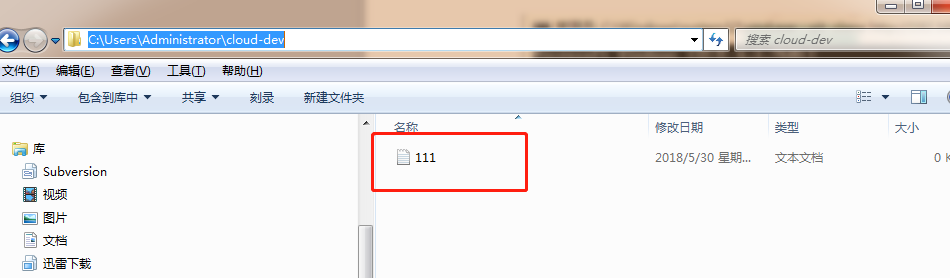
git config --global user.email ["admin@example.com"](mailto:\"admin@example.com\")

克隆仓库到本地

git clone <http://192.168.226.131/cloud-dev/cloud-dev.git>



我们在此目录创建一个文件，测试代码提交：



在命令行进入项目目录

cd cloud-dev

用dir命令查看目录内容



我们将111.txt上传到gitlab仓库。

添加文件

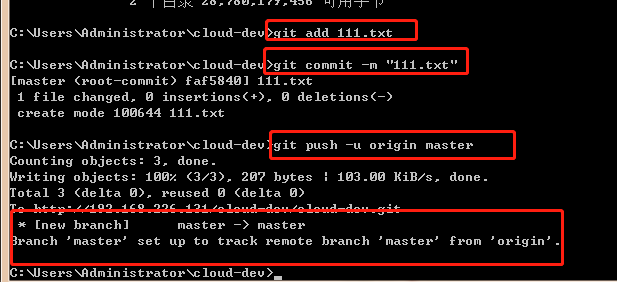
git add 111.txt

添加描述信息

git commit -m "111.txt"

上传文件到master分支，如果没有自动会创建。

git push -u origin master



用命令查看当前项目分支情况信息，看到有了一个master的分支，此分支为主分支，只有管理员有权限对其进行操作。

git branch



下面创建develop分支

git branch develop

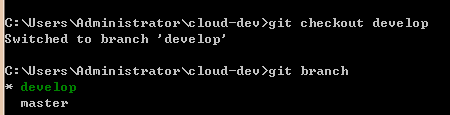


切换分支

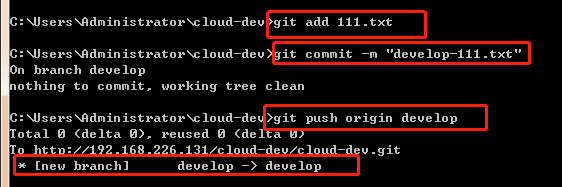
git checkout develop

查看分支

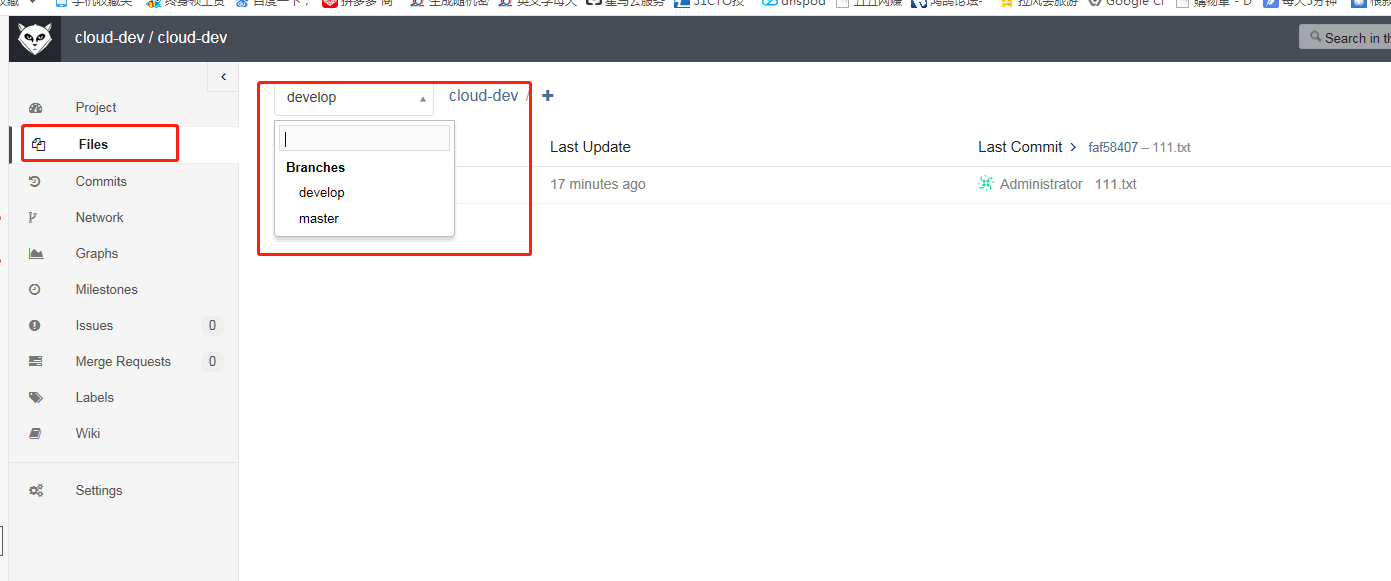
git branch



提交分支改变信息。



提交完成后我们在项目里可以看到：

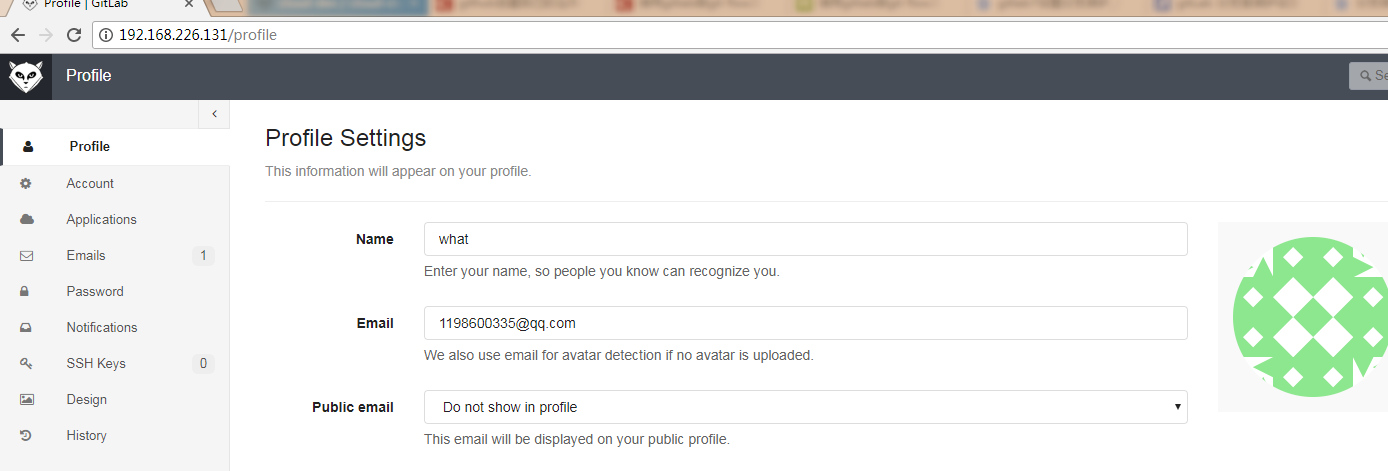


注意：分支必须开启protect保护。

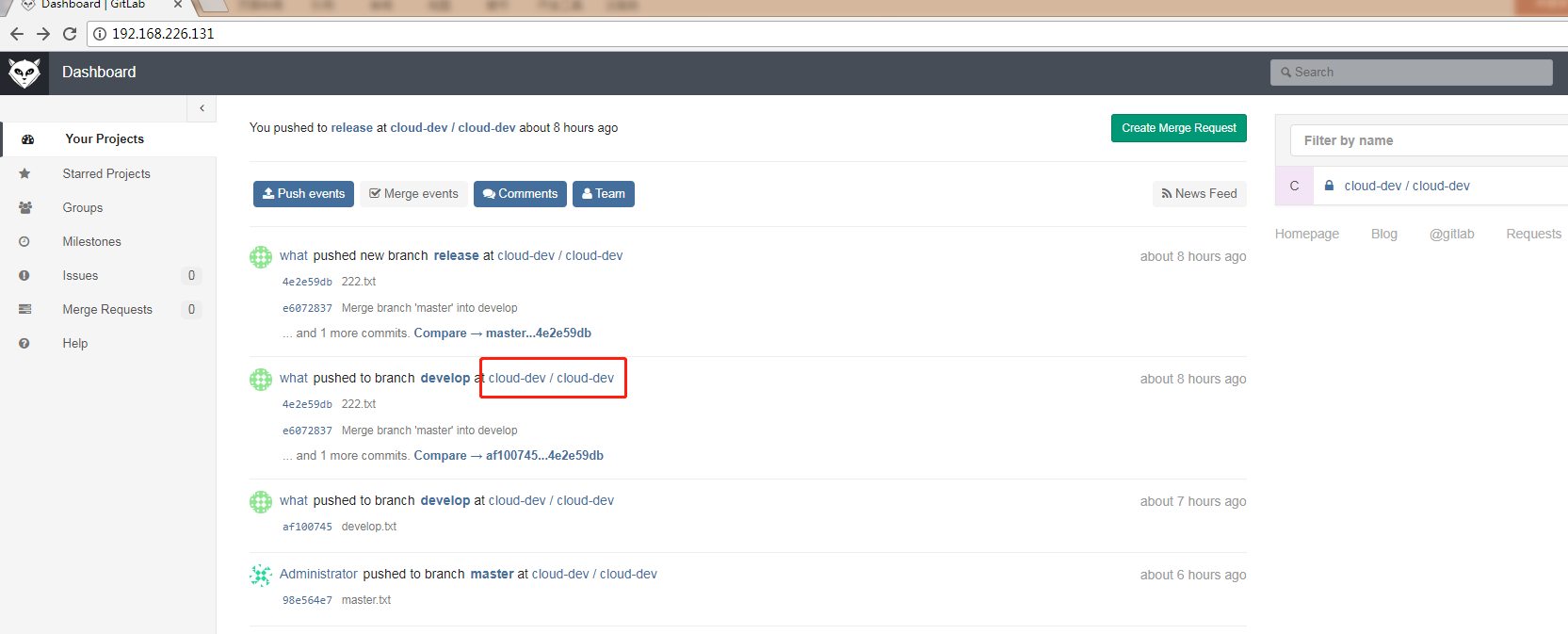
1. 分支合并申请。

开发人员what在develop分支进行代码开发工作，当代码开发完毕需要合并分支时，提交develop分支合并到master分支的请求，当管理员审核通过后方可合并分支。

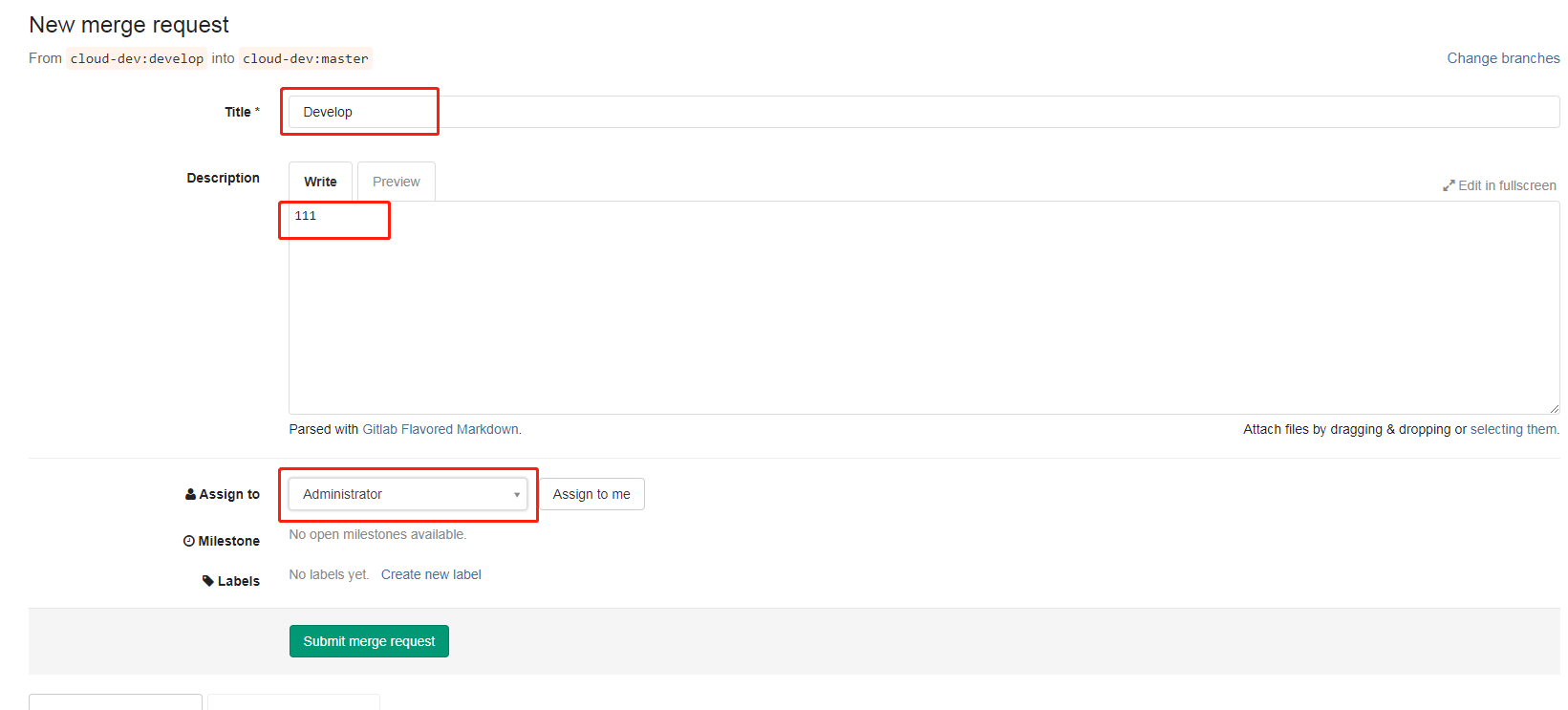
使用what账户登陆gitlab



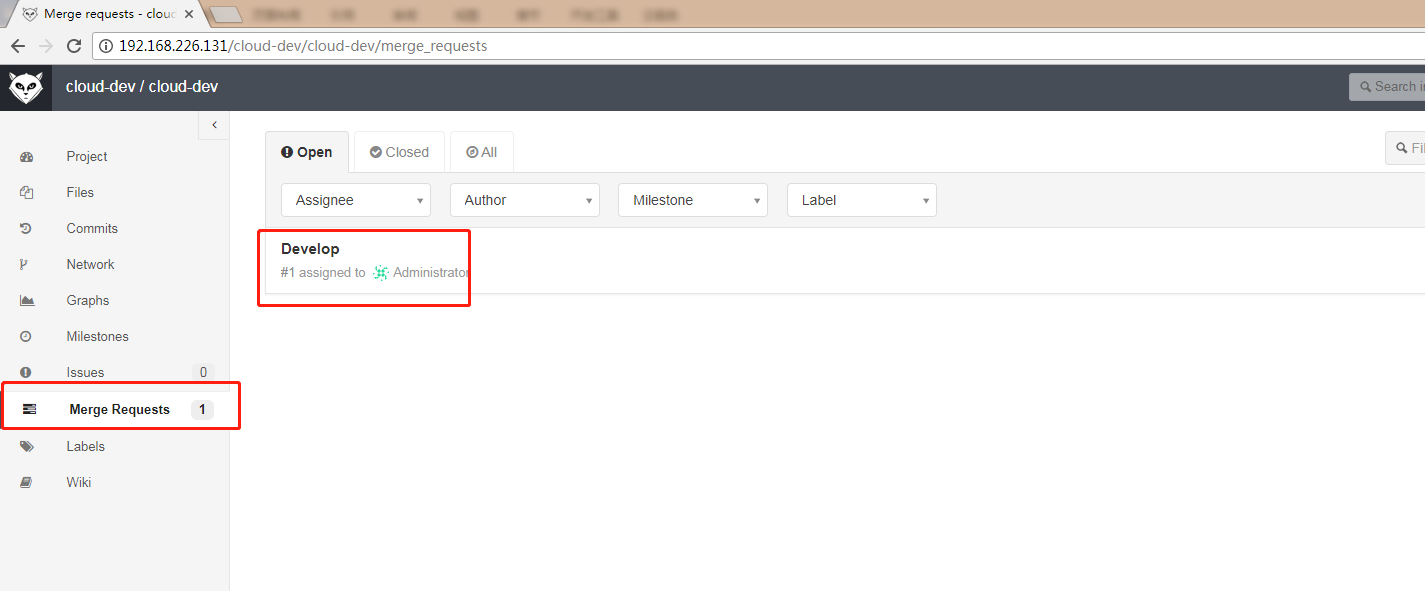
在项目界面，提交合并申请，选择相应的分支，然后提交合并申请。



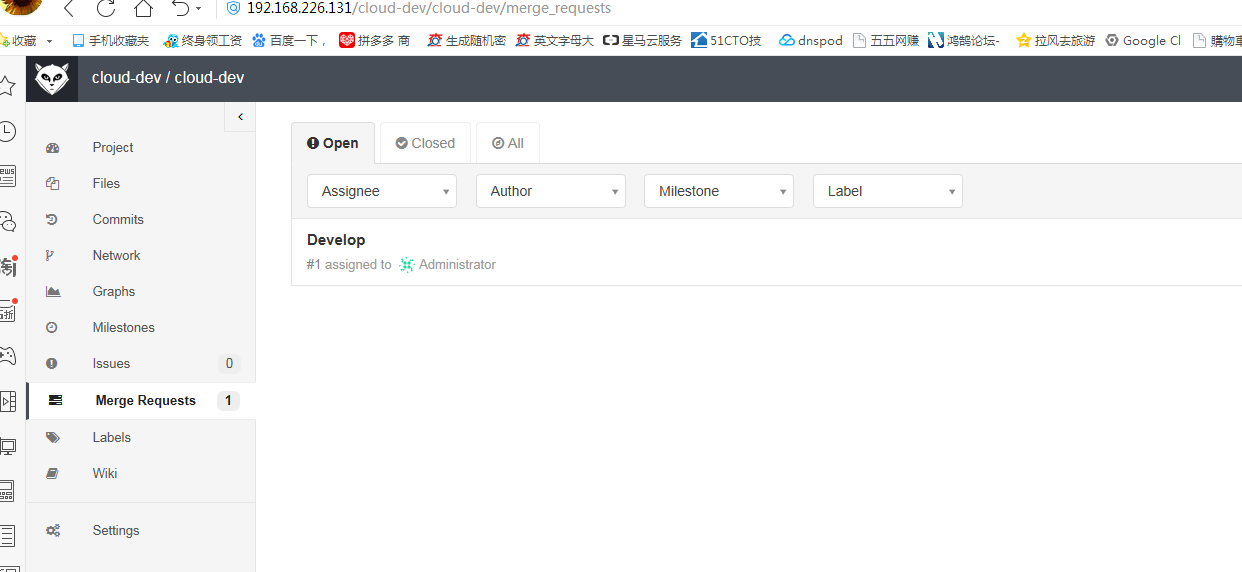
创建合并分支请求



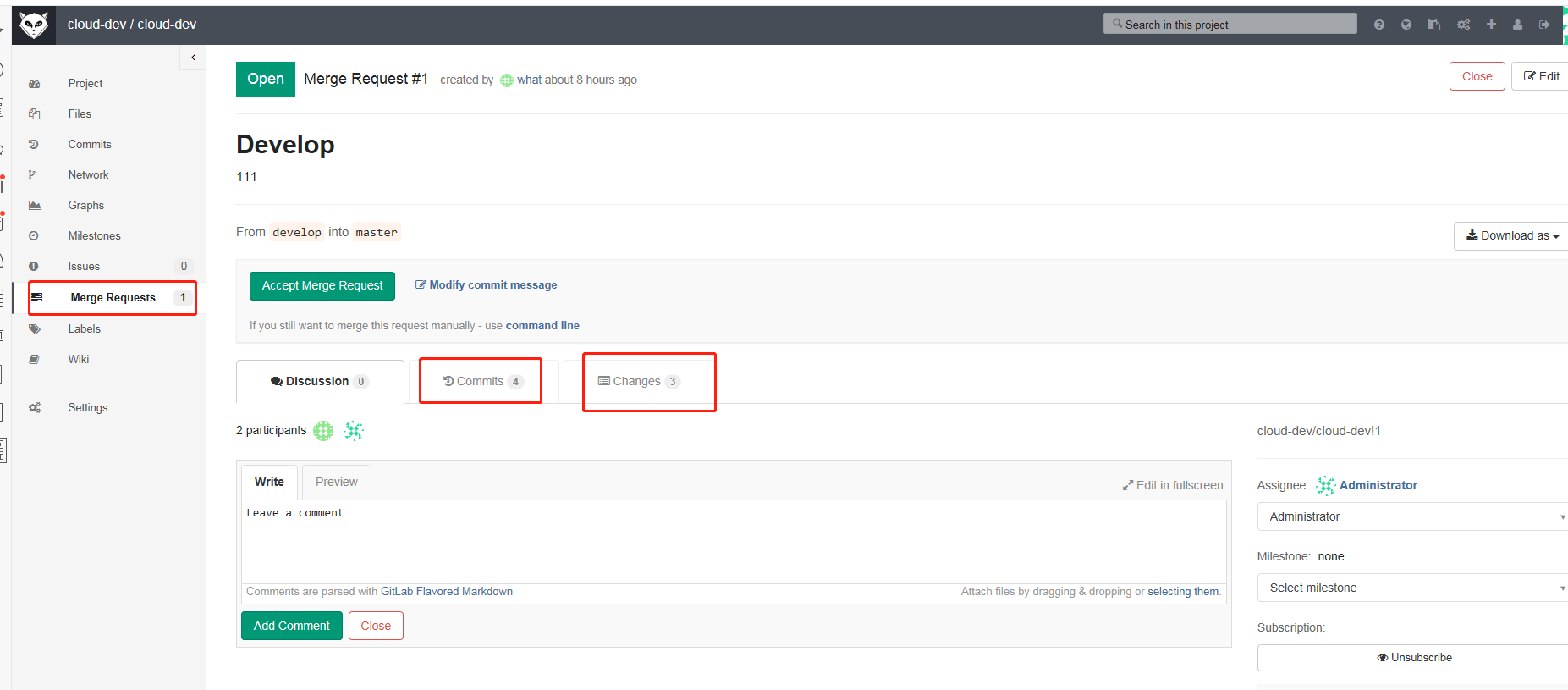
提交分支请求后，在这里可以看到：



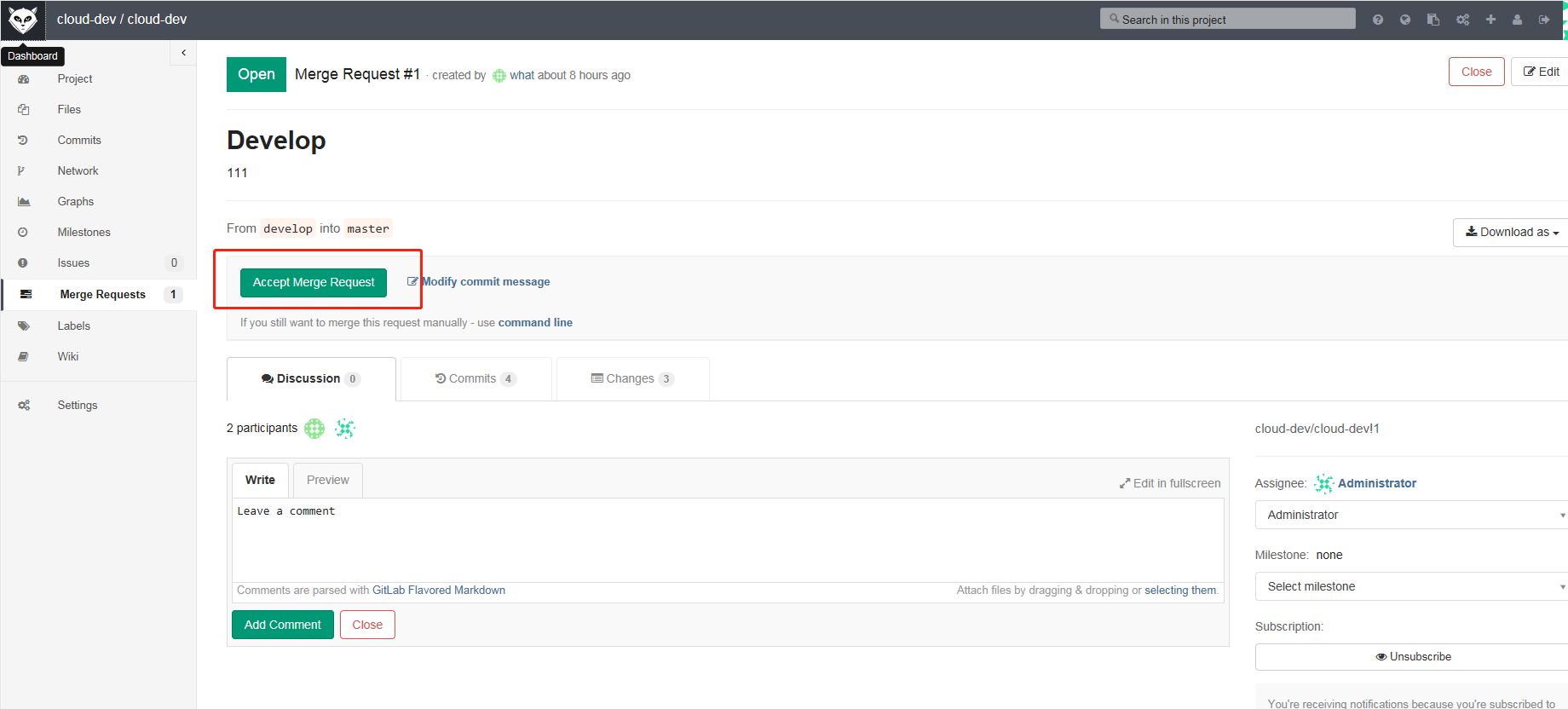
我们来到管理员账户的gitlab中，请求已经到达：



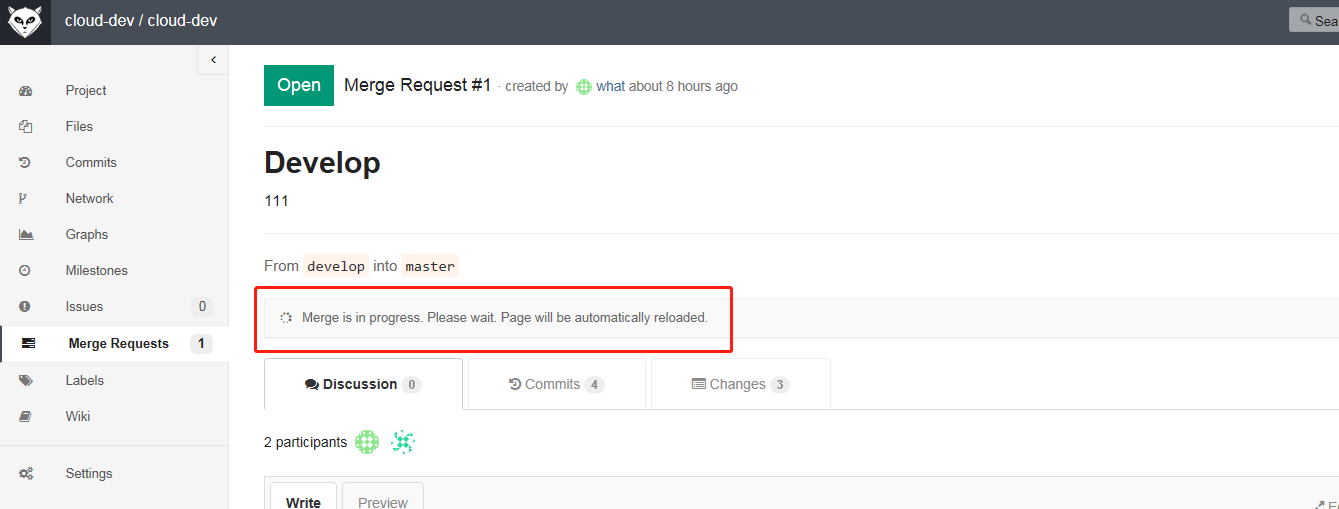
我们进去查看请求，可以看到合并申请的内容以及申请合并的用户。代码审查，不光是查看内容，要实际测试后没有问题，在同意开发着的合并请求。



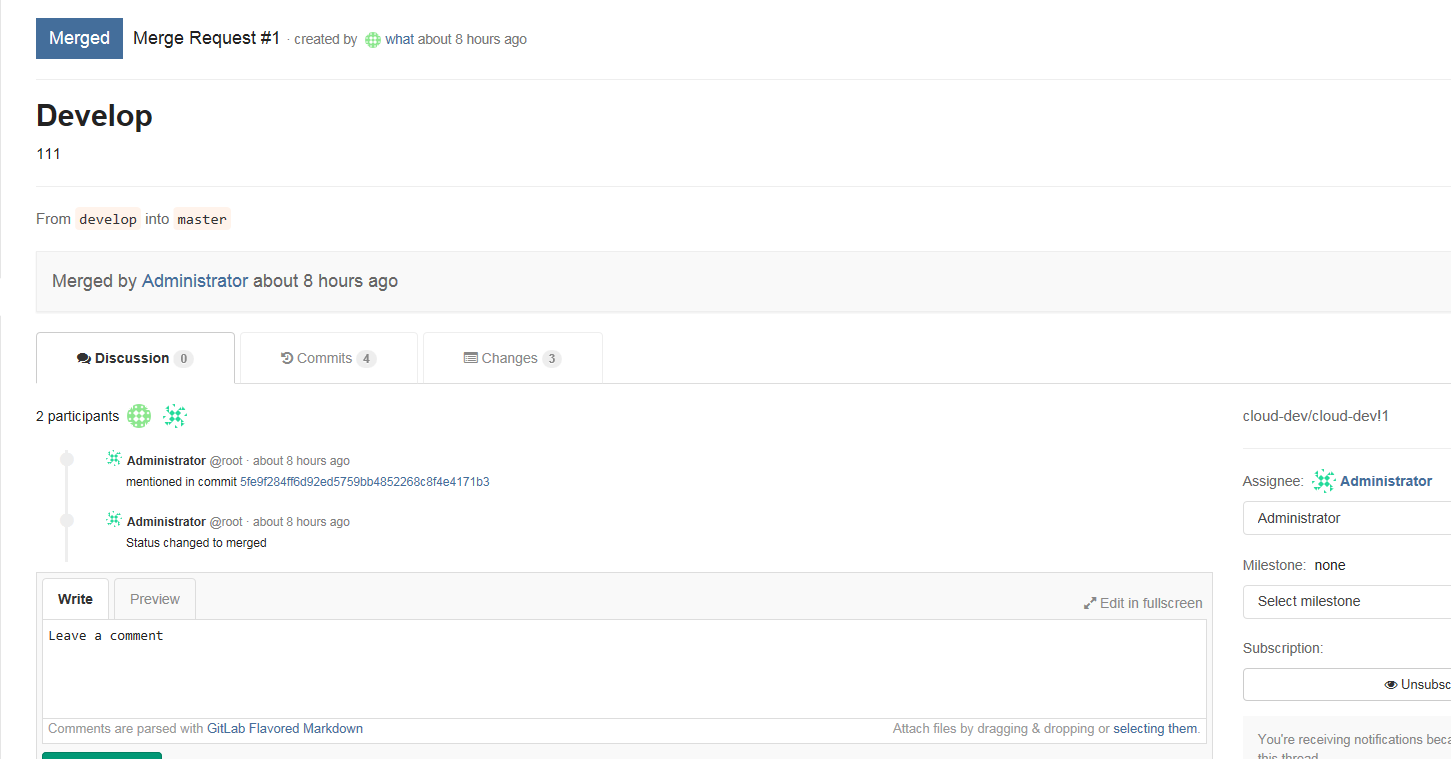
当管理员审查完文件内容或者查看了代码之后，选择同意合并申请。同意后，分支代码会合并到master主干里。



开始合并



合并完成



合并完成后查看master分支内容，发现develop所有内容都已经合并到master分支

