

# git 和 BUG 管理系统，持续集成环境 安装和学习报告

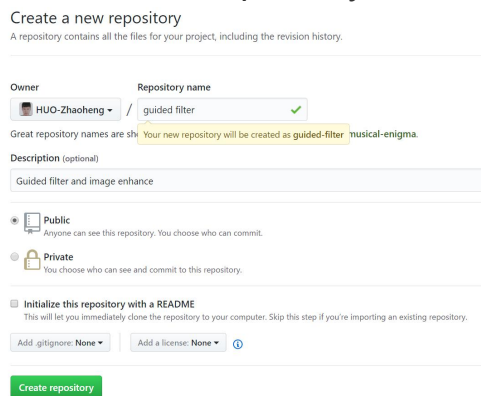
## 一，Github

github 是一个基于 git 的代码托管平台，付费用户可以建私人仓库，我们一般的免费用户只能使用公共仓库，也就是代码要公开。

如今有两种方式可以使用 Github，一种就是通过网页直接上传，不用记一大堆的命令，第二种比较麻烦但是一劳永逸，学会用本地客户端来上传，但是要记命令。下面介绍第一种：

### 1. 注册

### 2. Create a new repository:输入你的项目名称和描述，中文貌似不可以用

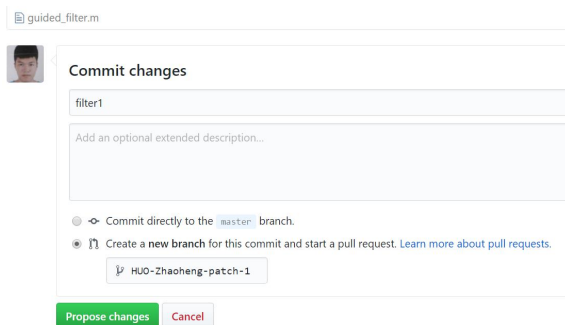


### 3. 上传文件

点击 Upload

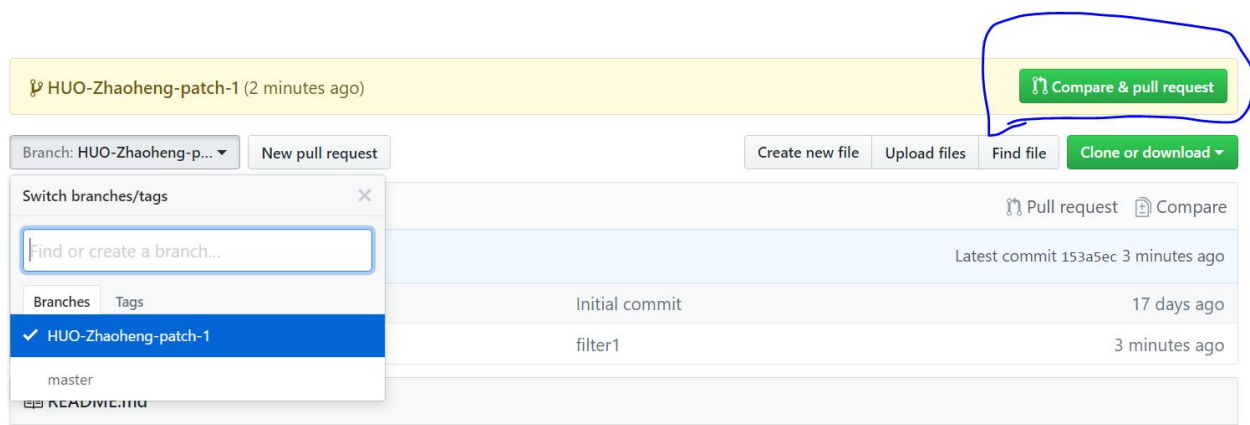
上传文件

此时有两种选择，一种是直接 commit 到 Master，第二种是创建一个新的 Branch



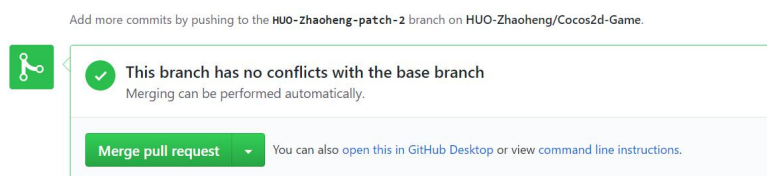
这里为了体验 branch，我们直接选择第二个。（上传文件需要你文明上网，否则你会卡住）

### 4. 合并 branch

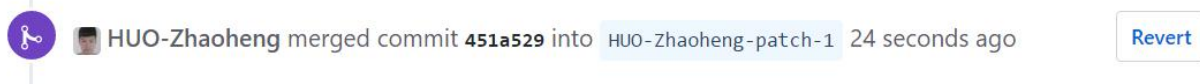


为了体验效果这样，我们 branch1 传一个文件，在 branch2 传另外一个文件，看看合并的时候发生什么。

## 5. 确认合并



注意：github 提供回溯功能，如果对合并不满意，还可以撤回



如果不想要 branch2 就可以手动删除了，这里我们手动删除了。

可以看到下面结果：

guided_filter.m	filter1	14 minutes ago
main.m	branch 2	7 minutes ago

来自 branch2 的文件合并到 branch1 中。

那如果两个 branch 含同名文件怎么办呢？

重新创建 branch2，我们把一个代码的第一行删去，重新上传。

将 Branch2 合并到 branch1.

新的 branch1 的代码里没有第一行

```
1
2  Img=imread(image_name);
3  p=imread(image_name);
```

可以看到 branch2 中的文件覆盖了 branch1 的文件

合并报告里 filed changed 是这么写的：第一个行前面的减号代表这行被删去，第二行的加号代表多了一个空行。

```
-image_name='beauty_with_freckle.bmp';
```

```
+
```

```
Img=imread(image_name);
```

最后将 branch1 合并到 master

## 二，BUG 管理系统

我们选择的是一款国产 web 管理云——蒲公英 Bug 管理云

我认为它有以下几个优势：

- 完全免费

现在免费，今后也一直免费。

- 网站采用云端部署

这也就意味着无需开发者自己搭建、部署，注册一个蒲公英账号即可，所有操作均在 Web 端完成。

- 系统使用 AngularJS 编写，简单快速易用

目前整站无刷新的效果归功于 AngularJS，用起来还是蛮爽的。

操作流程和功能最大限度的做到了简单易用

- 直观的问题系统

优先级、类型、状态等问题属性让你更直观连接问题的状态。同时你还可以随时添加评论并查看动态。

全面清晰的工作流程共有新建、处理中、已解决、未解决、待反馈、关闭六步流程，每一步工作流程都可以指派给你的团队人员，每一位成员的完成进度都是清晰可见的。



- 轻便的项目管理

你只需要一个蒲公英的帐号便可以管理项目，包括你创建的和参与。在这里项目的统计、动态、成员和问题都可以一览无余。



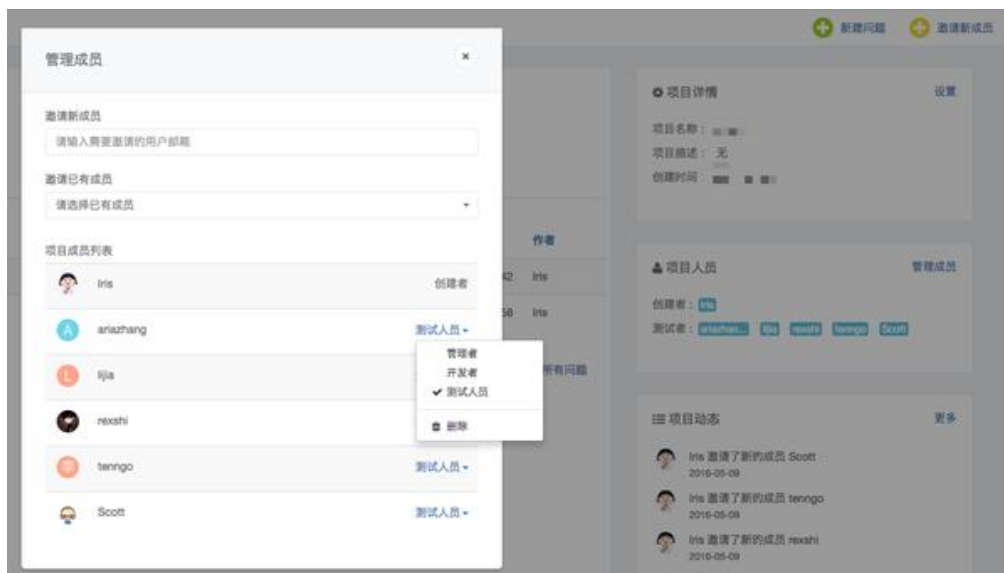
### • 文件共享

你可以把 Bug 管理云当做项目网盘使用，上传文件不受格式和大小的限制。



### • 角色设置

管理员、开发、测试三种角色，让协作更加方便。



- 支持 Markdown 语言

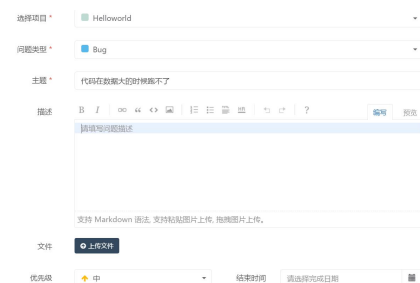
在 Bug 管理云中，问题的描述文本框内是支持用 **Markdown** 来编写的，在开发的时候尝试用过几款市面上已有的 Markdown 编辑器，发现都不太合适。所以，你现在看到的 **Bug 管理云** 使用的编辑器是我们自己写的。

- 沿用蒲公英一贯的简洁风格

我们以简单清晰为前提，通过高辨识度但不刺眼的颜色区分问题的类型和状态等信息

学习报告：

创建问题：



每个登陆上去的人都可以看到整个团队的问题。

问题状态有：



### 三，Jenkins 持续集成环境

Jenkin 的前身是 Hadson，因为被 Oracle 收购了，所以换个名字继续开源，这个有点像 MySQL。持续集成总是跟敏捷开发相关联，其实它就是一个后台服务+web 管理配置页面，它可以自动化（定时或事件触发）地执行某项任务，比如编译程序、打包程序、自动发布等等。这个在 web 开发或者大项目的多人合作上面很有帮助。只要配置好了，然后可以个人做个人的，Jenkins 会自动的从 svn 或 git 上面获取最新的代码，整合编译发布。相当于主程的很大一部分工作（发布版本）都由 Jenkins 自动完成了。因为手游项目比较小，所以 Jenkins 的意义不是非常大，但是通过 Jenkins 依然可以自动化的处理资源、整合发布版本、自动化处理渠道包等等。

#### 1、搭建 Jenkins 环境

从 <http://jenkins-ci.org/> 下载对应的版本安装，它会自动启动一个后台服务。在浏览器中访问 <http://localhost:8080/> 可以管理和配置 Jenkins，这个地址同样是可以设置的。

在"系统管理"里面可以设置全局选项（比如 svn 版本），也可以下载和更新插件。

搭建环境很简单，这个也是 Jenkins 比其他持续集成工具要优秀的地方。

#### 2、新建一个任务（Job）

常用的是自由风格的软件项目，而多配置项目用于这样的情形：任务相似，但是一些选项和配置不同，比如 debug 和 release 版本，比如 android 的各种渠道包，这个就可以添加对应的 Key-Value 来进行配置，构建的时候就可以选择对应的内容来构建，而不是全部构建。

选择仓库

☐ None  
☐ CVS  
☐ CVS Projectset  
☒ Subversion  
Modules

Repository URL

**Repository URL is required.**

Credentials  [http://blog.csdn.net/langresser\\_king](http://blog.csdn.net/langresser_king)

Local module directory

Repository depth

Ignore externals ☐

在 Repository URL 里面配置 svn 的服务器地址，Credentials 里面配置用户和密码。如果是老的 svn 插件可能没有 Credentials，推荐更新到最新。

### 构建触发器

- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ Build when job nodes start
- ☐ Poll SCM
- ☐ [BuildResultTrigger] - Monitor build results of other jobs
- ☐ [FSTrigger] - Monitor files
- ☐ [FSTrigger] - Monitor folder
- ☐ [IvyTrigger] - Poll with an Ivy script
- ☐ [ScriptTrigger] - Poll with a Groovy script
- ☐ [ScriptTrigger] - Poll with a shell or batch script
- ☐ [URLTrigger] - Poll with a URL

### 构建

增加构建步骤 ▼

### 构建后操作

增加构建后操作步骤 ▼

"构建触发器"里面可以设置这个任务如何触发（比如按时间触发、其他任务完成后触发等等，这个后面会重点说明），插件里面有很多触发类型可选，比如[FS Trigger]这个就是一个插件的功能。

"构建"里面可以设置如何完成构建。可以是 windows 的命令，也可以是 shell 脚本，shell 脚本里面同样可以是 python 代码。这些在 windows 下面都是可以使用的。

"构建后操作"可以设置构建完成后邮件通知或者是其他事情。通过插件，可以在这里完成更多的功能。

### 3、开始构建



点击"立即构建"则可以立即开始执行构建流程。如果在配置里面设置了 Poll SCM，则这里还会有 Polling Now 的选项。Poll 可以查看 svn，如果有更新，则开始构建，否则不做任何处理。这个后面实际应用里面会介绍。

"工作空间"里面可以看到 Jenkins 检出的 svn 文件夹内容（一般在 'Jenkins 安装目录' /jobs/任务名/workspace 文件夹下）。

### 4、查看构建结果





在这里可以查看具体的构建结果。svn 的更新内容、命令行的输出都会在这里显示出来（比如 Python 代码的 print）。不过我使用的时候有一个问题就是这里的显示结果很诡异，通过某种方式点进来就会显示 python 的输出，而另外某种方式就不会。