这篇[文章](http://www.51testing.com/html/93/javascript:;" \t "http://www.51testing.com/html/93/_self)比较简单，适合初学持续集成的读者，本文可以帮助你对基于[Jenkins](http://www.51testing.com/html/93/javascript:;" \t "http://www.51testing.com/html/93/_self)的持续集成有一个比较全局的概念。

**提出问题**

　　为了使用HTTPS，我将博客从GitHub Pages迁移到了我的服务器上。由于博客基于Hexo，在迁移之前，我的工作流程是：

|  |
| --- |
| 本地写Markdown格式文章->Hexo生成HTML并推送到GitHub->GitHub Pages自动使用最新内容 |

　　现在由于Hexo渲染以后的HTML文件需要放到服务器上，那么工作流程变为：

|  |
| --- |
| 本地写Markdown格式文章->Hexo生成HTML并推送到GitHub->手动登录服务器  ->进入放博客文章的文件夹->执行命令git pull |

　　增加的两步虽然说操作量不大，但是总显得很麻烦。于是我希望，在我把博客的HTML文件push到GitHub以后，服务器能自动从GitHub上把HTML内容拉下来。由于使用了Nginx，所以只要博客的HTML发生了更新，那么使用[浏览器](http://www.51testing.com/html/93/javascript:;" \t "http://www.51testing.com/html/93/_self)访问https://kingname.info时，新内容自动就会出现。这样一来，对我来说，看起来就像是我刚刚push了网页内容到GitHub，博客就自动更新了。

　　让服务器从GitHub上面拉代码，这个操作本身很简单。人来操作就一行代码git pull。写个[Python](http://www.51testing.com/html/93/javascript:;" \t "http://www.51testing.com/html/93/_self)脚本也就两行代码。但问题是，服务器怎么知道GitHub发生了更新？

　　有人说，Kingname，你不是做爬虫吗？你写个爬虫，每一秒检查一下GitHub不就可以了吗？这种办法当然可以。但问题是，我一周就更新一次博客，但这个爬虫为了等这一次更新，一周要访问GitHub高达604800次？更何况我有时候一个月都不更新。

　　所以显然不能让服务器主动检查GitHub更新，这种“轮询”操作效率太低。那么如果反过来，一旦GitHub有更新，它就通知服务器，然后服务器再去拉代码，这不就简单高效了吗？

**解决问题**

　　为了实现这个目的，就需要使用一些持续集成的工具。本文使用的是Jenkins。

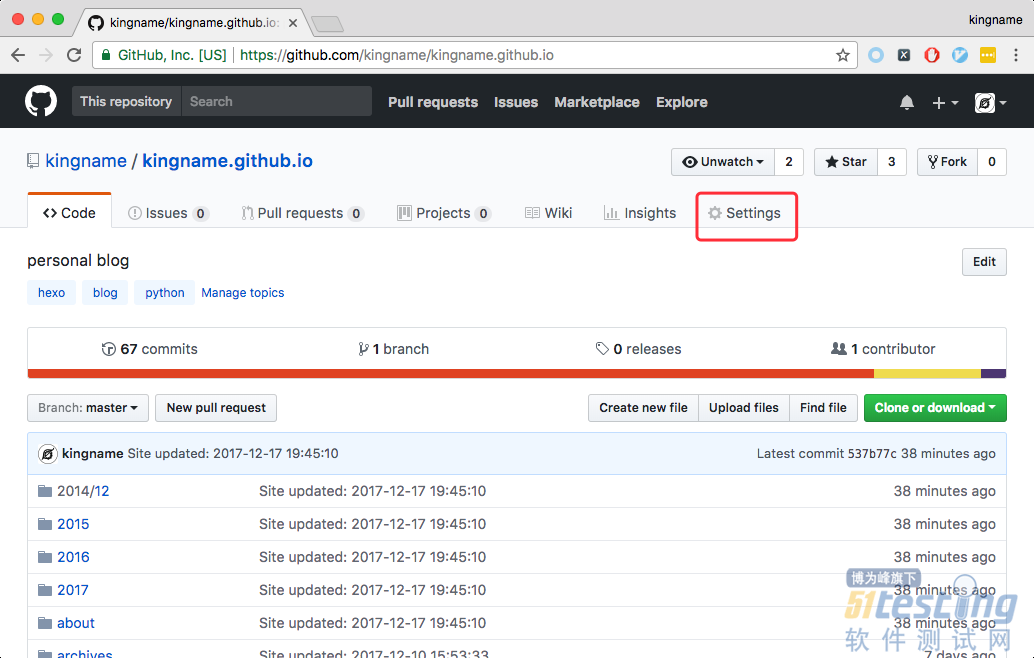
　　Jenkins在敏捷开发界可以说是大名鼎鼎了。使用Jenkins搭建一整套持续集成环境，可以实现开发者往代码仓库一提交代码，代码自动进行[单元测试](http://www.51testing.com/html/93/javascript:;" \t "http://www.51testing.com/html/93/_self)，覆盖率测试，代码风格检查自动生成报告，自动通知部门同事开始Code Review。当代码被合并入主干以后，服务器自动拉下最新代码，自动编译，自动在几千几万台服务器上部署。在整个过程中，开发者只需要做一件事，那就是git push（当然在实际情况下，后面还会加一些参数）。

　　本文实现的是博客的自动部署，没有任何测试，也没有Code Review，也没有编译，服务器也只有一台，复杂程度当然远远低于持续集成。不过管中窥豹，来看看这个简单地流程是如何走通的，对初学者也会有帮助。

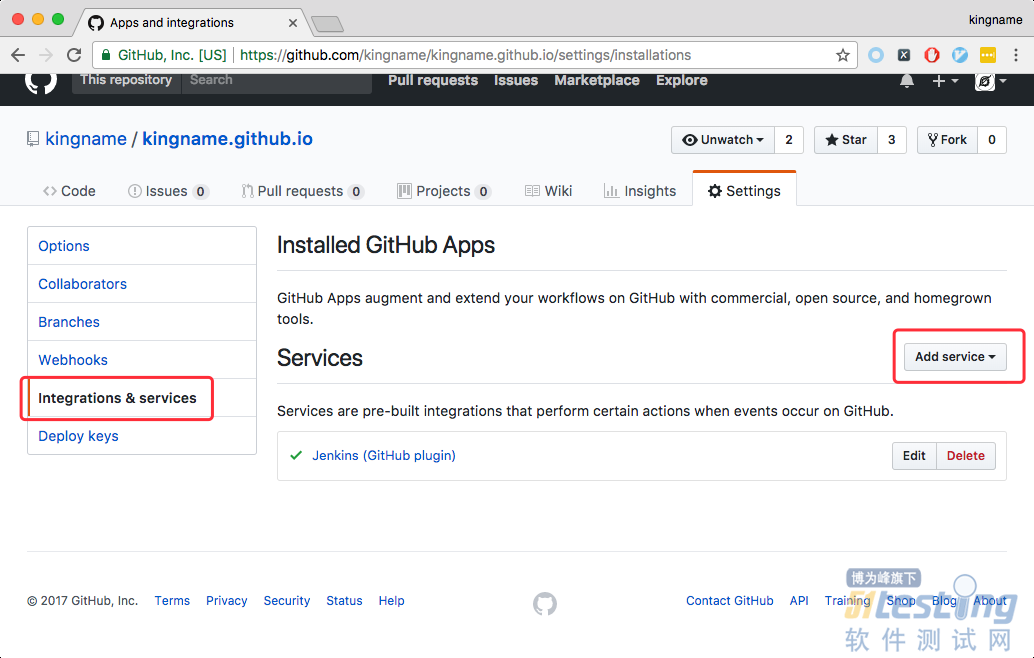
　　本文假设你已经在服务器上面搭建好了Jenkins环境。如果你还没有安装Jenkins或者不会安装，那么可以“参考”这篇文章：搭建持续集成环境(一)。由于这篇文章是18个月以前写的，系统也是小众的Arch [Linux](http://www.51testing.com/html/93/javascript:;" \t "http://www.51testing.com/html/93/_self)，所以建议你还是在网上搜一下最新的Jenkins安装教程比较好。

**设置GitHub**

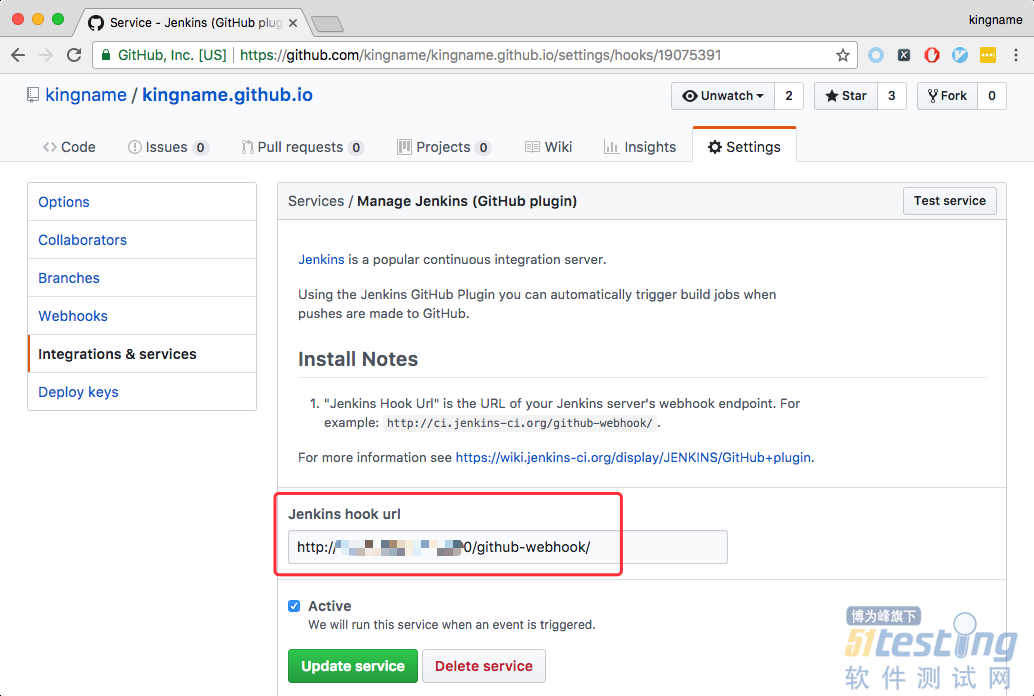
　　在GitHub中进入博客所在的Repo，并点击Settings，如下图所示。

[](http://www.51testing.com/batch.download.php?aid=79140)

　　在设置页面，单击左侧的Integrations & services，并选择Add service，如下图所示。

[](http://www.51testing.com/batch.download.php?aid=79141)

　　从下拉菜单中，选中Jenkins (GitHub plugin)。在新打开的界面，填写Jenkins的信息，如下图所示。

[](http://www.51testing.com/batch.download.php?aid=79142)

　　其中Jenkins hook url填写的是我的服务器的Jenkins地址加上/GitHub-webhook/，所以完整的地址为http://xx.xx.xx.xx:8080/GitHub-webhook/。把这里的xx换成实际的IP地址或者域名即可。需要注意的是，网址末尾的斜杠一定不能省略。

　　填写好信息以后保存，GitHub就配置好了。

**配置Jenkins**

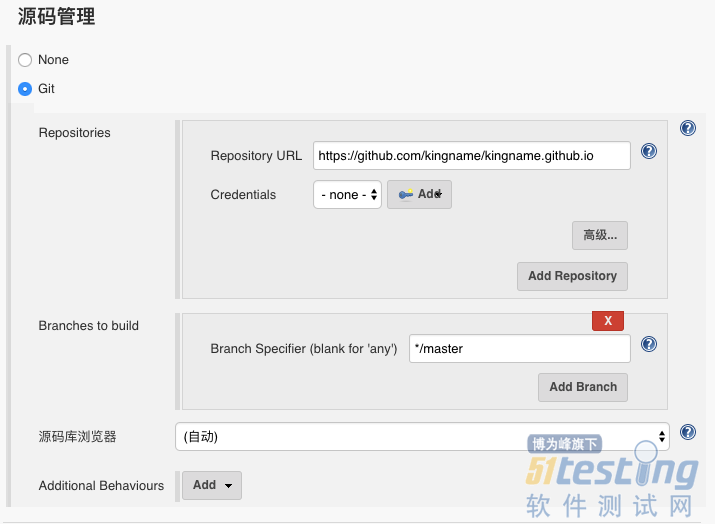
　　Jenkins需要安装GitHub Plugin才能接收到GitHub发来的通知。进入Jenkins的系统管理-管理插件，在可选插件选项卡中，搜索GitHub plugin并安装，安装完成Jenkins会重启。重启以后可以在已安装选项卡下面发现GitHub plugin，如下图所示。

[](http://www.51testing.com/batch.download.php?aid=79143)

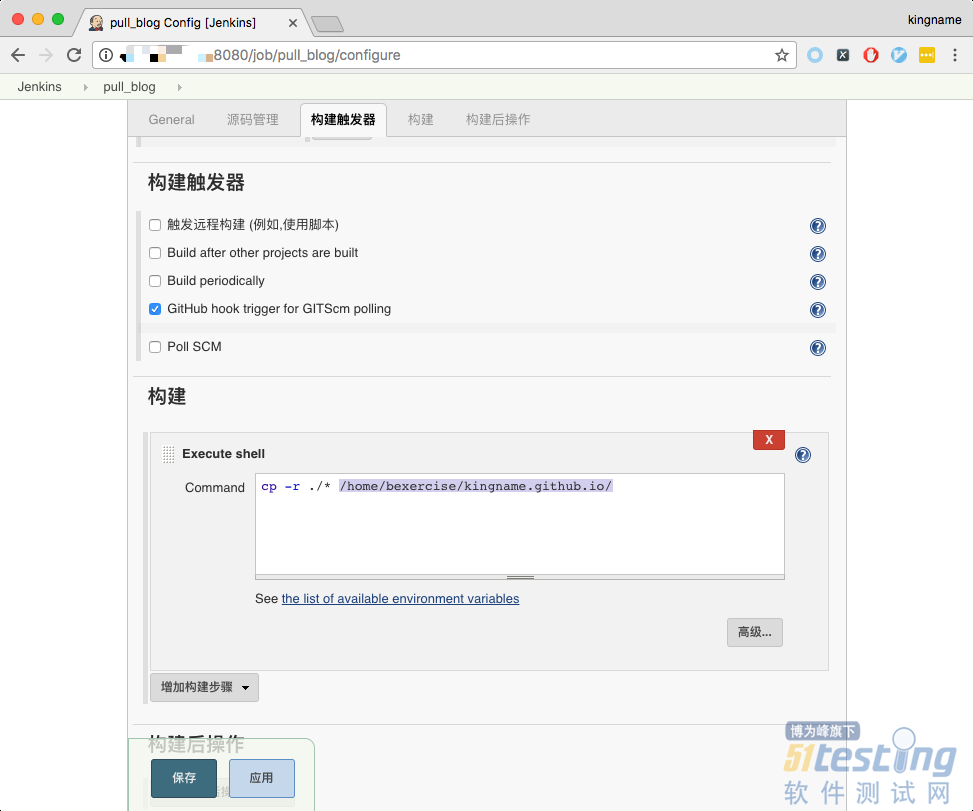
　　回到Jenkins首页，单击左上角新建按钮，新建一个项目，项目类型为构建一个自由风格的软件项目。任务的配置信息如下图所示。

[](http://www.51testing.com/batch.download.php?aid=79144)

　　其中的项目名称和描述可以随意填写。勾选GitHub project，并把博客对应的GitHub Repo的地址填入。往下拉，看到源码管理，点选Git，依然填写博客对应的Repo地址。如下图所示。

[](http://www.51testing.com/batch.download.php?aid=79145)

继续往下拉，在构建触发器单击增加构建步骤，在弹出的下拉菜单中选择Execute shell。勾选GitHub hook trigger for GITScm polling。在Execute shell对应的输入框中输入命令，将当前目录下的所有文件和文件夹全部复制到/home/bexercise/kingname.github.io/文件夹下。如下图所示。

[](http://www.51testing.com/batch.download.php?aid=79146)

　　其中，文件夹/home/bexercise/kingname.GitHub.io/里面的内容如下图所示。这是Hexo生成的HTML文件和资源文件。

[](http://www.51testing.com/batch.download.php?aid=79147)

　　这里需要解释一下这一条命令：

　　cp -r ./\* /home/bexercise/kingname.GitHub.io/

　　其中的cp -r表示复制文件和文件夹。./表示当前目录。./\*表示当前目录下面的所有内容。因此整条命令的意思是把当前目录下的所有内容全部复制到/home/bexercise/kingname.GitHub.io/下，并且如果文件名相同，就会直接覆盖。需要注意的是，kingname.GitHub.io这仅仅是一个普通的文件夹而已，别看它的名字长得像个网址，但其实它只是一个名字比较怪的普通文件夹而已，没有什么特殊的意义。

　　这个项目在构建的时候，它会自动从GitHub上面对应Repo所有的文件拉取到当前的文件夹下，所以执行了这一条复制命令以后，博客HTML文件自然就被复制到了网站的根目录下。

**修改权限**

　　由于Jenkins在安装的时候，会自动创建一个名为jenkins的普通账号，这个账号没有管理员权限。jenkins执行命令的时候，它也会使用这个账号。但是由于kingname.GitHub.io这个文件夹是用户bexercise创建的，所以jenkins账号默认是没有权限读写这个文件夹的。现在需要给jenkins账号授予权限。使用bexercise这个账号登录服务器，使用以下命令给jenkins赋予权限，让它可以读写kingname.GitHub.io文件夹：

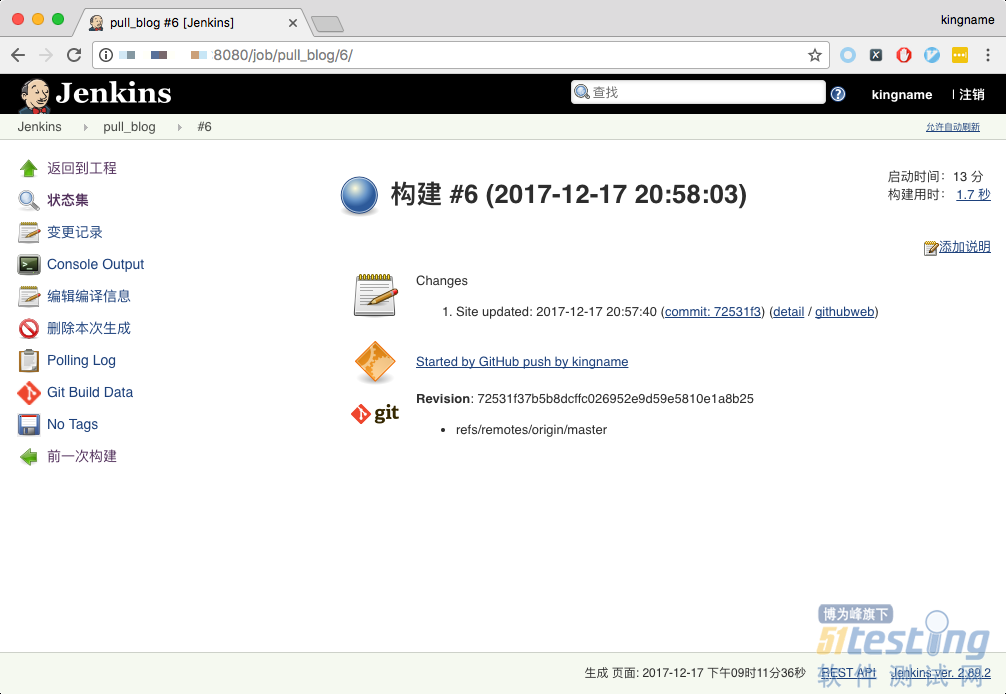
　　sudo chown -R jenkins:jenkins /home/bexercise/kingname.GitHub.io

　　执行完成这一行命令以后，jenkins才可以把其他地方的文件复制到这个文件夹里面。

**使用方法**

　　没有什么复杂的使用方法，在Hexo中，执行命令hexo d就可以把本地生成好的HTML文件提交到GitHub中。然后打开浏览器，打开博客，发现新的文章已经出现在首页了。

　　在Jenkins项目的执行历史里面，也可以看到它被自动触发而产生的历史记录。如下图所示。

[](http://www.51testing.com/batch.download.php?aid=79148)

**举一反三**

　　由于Jenkins可以运行Shell命令，进行单元测试本质上也是运行一条命令，那这不就可以实现自动进行单元测试了吗？那么如果把Shell命令改成运行一个Python脚本，那不就可以做任何事情了吗？如果Python脚本里面写了发送邮件的代码，那不就实现了你一提交代码，其他人就收到邮件了吗？