**SVTI持续集成环境**

**用户指南**

目录

[前言 1](#_Toc425001435)

[1. 简介 1](#_Toc425001436)

[1.1 持续集成是什么 1](#_Toc425001437)

[1.2 工作流程 1](#_Toc425001438)

[2. 使用方法 2](#_Toc425001439)

[2.1 准备工作 2](#_Toc425001440)

[2.2 提交代码 3](#_Toc425001441)

[2.2.1 命令行界面提交代码 4](#_Toc425001442)

[2.2.2 TortoiseGit界面提交代码 5](#_Toc425001443)

[2.3 结果通知 9](#_Toc425001444)

[2.3.1 Web界面 9](#_Toc425001445)

[2.3.2 邮件通知 9](#_Toc425001446)

[3. 常见问题与错误 10](#_Toc425001447)

[3.1 有人正在上传 10](#_Toc425001448)

[3.2 需要更新代码 11](#_Toc425001449)

[3.3 难以消除的CA告警 11](#_Toc425001450)

[3.4 上传分支错误 12](#_Toc425001451)

[3.5 请运行Git\_AutoCA.exe 12](#_Toc425001452)

[3.6 服务器当前分支错误 13](#_Toc425001453)

[3.7 Web页面出现错误调试信息 13](#_Toc425001454)

[3.8 邮件内容有误或乱码 13](#_Toc425001455)

# 前言

为了能在团队搭建一个和公司相似的开发环境，更好的管理代码，保证主线代码永远是OK的。于是耗时大半个月，写了一些脚本和C程序，终于实现了与公司基本类似的持续集成开发环境，并不断地改进和添加一些新特性。欢迎大家反馈各种BUG（最好把屏幕输出的调试信息截图保存），或者想添加一些实用小功能，可以向我提意见或者发到我的邮箱。

# 简介

## 持续集成是什么

关于持续集成，给我的感觉就是开发人员要经常性地把把代码合入主线，每次上传代码都会触发构建，然后服务器对代码进行自动化测试，测试通过并审核之后代码才能合入主线。

持续集成的好处很多文献资料都有说明。持续集成的关键是完全的自动化，读取源代码、编译、连接、测试，整个创建过程都应该自动完成。

## 工作流程

用户提交代码到

GIT服务器

CA语法检查

编译

成功:合入主线

失败:版本回退

CLT

邮件通知

图 1.1 SVTI持续集成环境工作流程

# 使用方法

由于持续集成的环境主要是在服务器搭建，所以客户端的使用是比较简单的。主要都是些Git的基本操作。

## 准备工作

1. 安装Git
2. 公钥认证

需要认证才有权限下载及合入代码，目前公钥主要是人工管理。

1. 右击桌面空白处打开Git Bash，输入ssh-keygen，然后一直按Enter键。

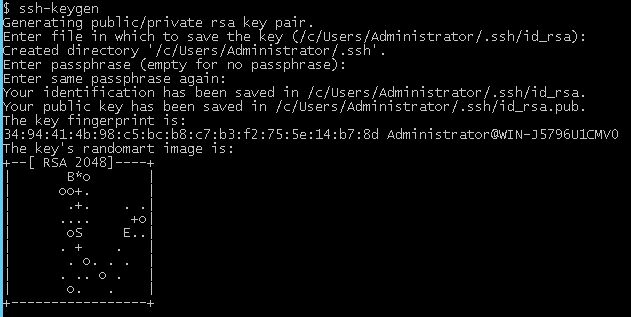


图 2.1 ssh-keygen命令结果

1. 把文件C:\用户\”你的用户名”\.ssh\id\_rsa.pub复制出来，重命名为自己的名字（如liangjinchao.pub）然后发送并同时通知管理员。
2. 设置邮箱

继续Git Bash里面输入以下命令(邮箱名和姓名写自己的)

git config --global user.email [liangjinchao@svti.com](mailto:liangjinchao@svti.com)

git config --global user.name liangjinchao

git config --global push.default simple

1. 下载代码

如果以上步骤成功，那么就可以Git clone代码了：打开git bash，切换到加密盘目录，然后输入命令：

git clone ssh://git@192.168.11.22/SVTI/Code

或者（如果装了Tortoise Git）：

在工作目录下右击Git Clone，地址栏写上面的ssh地址(ssh://….)，然后确定。

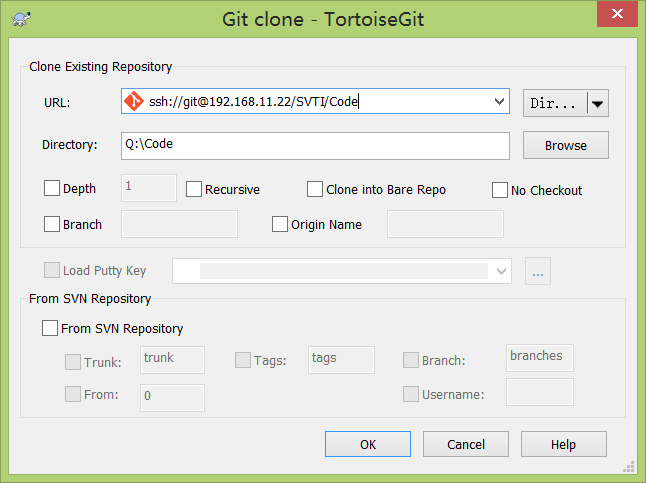


图 2.2 TortoiseGit下载代码界面

如果下载成功，恭喜你，公钥认证成功。如果失败，则检查前面几步是不是没做好，如果使用TortoiseGit检查是否按照图2.4的设置。

## 提交代码

提交代码的步骤和平常Git的操作一样，先commit（提交到本地）然后更新代码最后push（上传到服务器）。**唯一不同的是push的分支不是master，而是远程分支work**。当自动化测试通过后work分支会自动合入master分支。

### 命令行界面提交代码

命令行界面提交代码的相关命令如下：

1. commit到本地：

git commit --all

然后弹出来vi编辑器中，在文件开头填入提交信息，在将需要commit文件前面的#号去掉保存（ZZ）退出。

1. fetch+rebase更新代码:

git fetch origin

git rebase origin

rebase的过程中可能出现冲突导致失败。如果没有冲突则进行下一步。

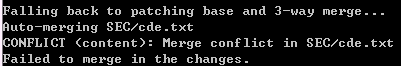


图 2.3 rebase出现冲突的例子

这时打开产生冲突的文件，找到出现“<<<<<<<”、“========”、“>>>>>>>”的地方，就是冲突产生的地方。修改这些地方的代码，其中“<<<<<<<”与“========”之间的部分是服务器的主线代码，“========”与“>>>>>>>”之间的部分是你提交的代码，两者产生冲突需要修改，且修改完不能含有原来的“<<<<<<<”、“========”、“>>>>>>>”，然后再次rebase:

git add --all

git rebase --continue

如果修改冲突时间过长，中途可能有人又合入了新的代码，则需再次进行步骤（2）。

1. push代码到服务器：

git push origin master:work

上述都是合代码最基本的Git命令操作。git命令还有很多高级的功能和用法，可以通过—help参数来查看。

### TortoiseGit界面提交代码

如果实在是不习惯用命令行的方式来提交代码，可以使用图形化界面的TortoiseGit，足以满足大多数基本操作。当然啦命令行的功能永远是最完整的。

使用Tortoise Git则提交代码步骤要经过三步：

* Commit:提交代码到本地仓库
* Fetch+Rebase:更新代码并与主线代码结合
* Push:上传代码到服务器

第一次使用TortoiseGit提交或下载代码前先检查下以下设置是否正确：右击桌面空白处TortoiseGit----Settings，按图 2.4所示设置。

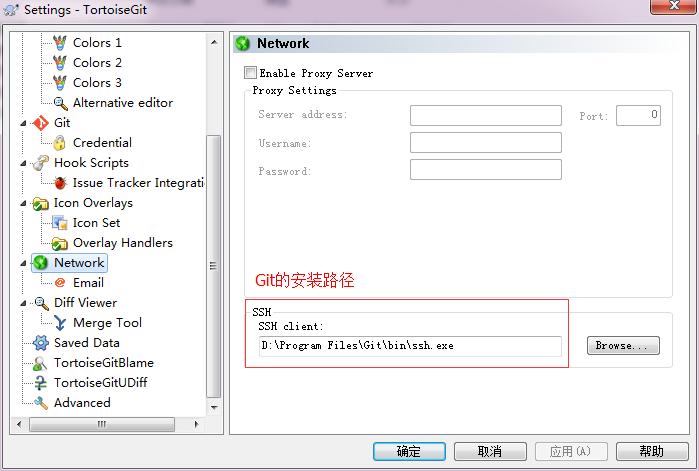


图 2.4 设置SSH客户端

TortoiseGit提交代码具体操作如下：

（1）Commit到本地：

右击仓库目录，选“Git Commit -> master”，然后输入提交信息，OK。

（2）Fetch+Rebase更新代码：

右击仓库目录，选Tortoise Git ---- Fetch 弹出窗口如图 2.5：

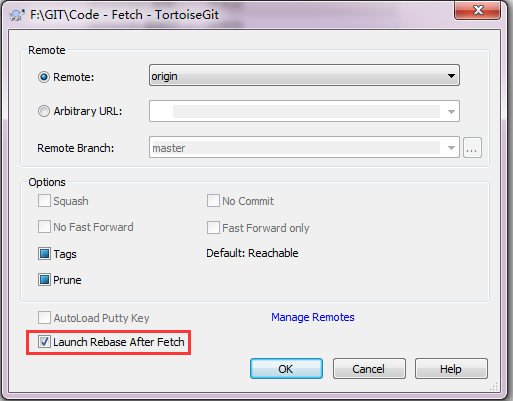


图 2.5 TortoiseGit的Fetch更新代码界面

勾选“Launch Rebase After Fetch”，然后点击OK。之后弹出的界面点Rebase。Rebase的过程可能产生冲突，比如你的代码和别人的改了同一个函数的同一行：

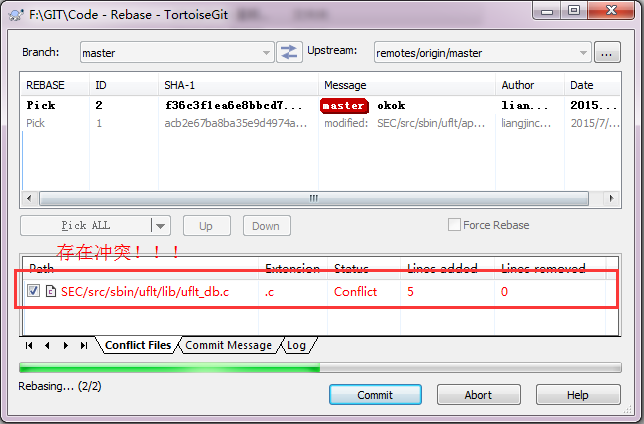


图 2.6 Rebase时产生冲突

如果存在冲突则双击图 2.6 中红色框起来的部分编辑冲突（界面如图 2.7，注意不要有多余的换行符）。冲突修改期间如果有人又合入了新代码，则修改完冲突之后需要再次进行步骤（2）。如果不存在冲突则进行下一步。

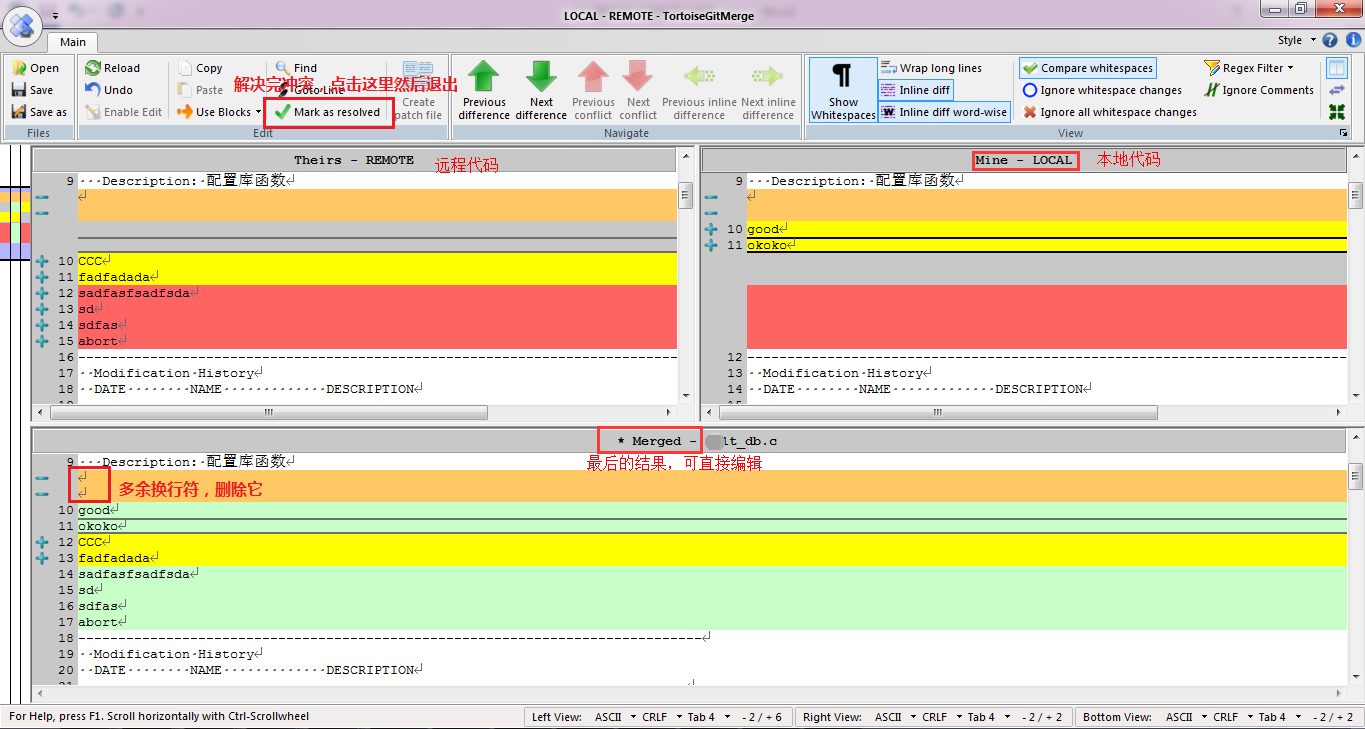


图 2.7 冲突编辑界面

（3）Push代码到服务器：

右击仓库目录，选Tortoise Git ---- Push弹出窗口如图 2.8:

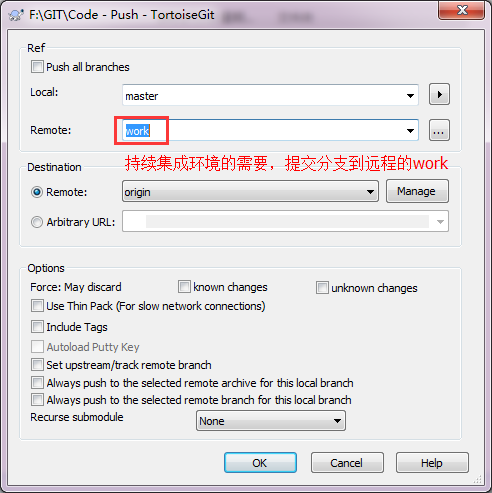


图 2.8 TortoiseGit上传代码到服务器

如果弹出图 2.9或图 2.10的界面则表明代码上传成功啦！当然还需要通过自动化测试才能合入主线（除非出现图 2.10的情况）。否则需要修改代码重新提交。

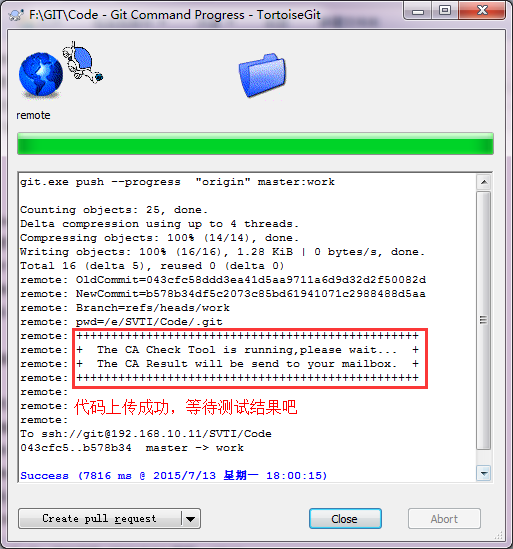


图 2.9 上传成功，需要等待测试结果

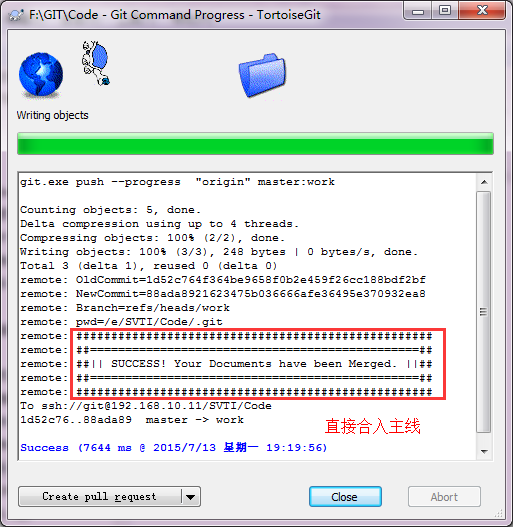


图 2.10 上传成功且直接合入主线

## 结果通知

### Web界面

打开以下链接实时地观察编译输出结果（有的任务可能还没开始）。

<http://svti:8080/> 或 <http://192.168.11.22:8080>

### 邮件通知

当自动化测试结束后，会收到邮件通知。如果收不到邮件，请检查2.1节中的步骤(3)进行操作。

# 常见问题与错误

当正常操作失败或屏幕打印奇怪的信息的时候，应及时截图保存错误信息，以备环境开发人员进行维护。下面列举了几种常见的问题。

## 有人正在上传

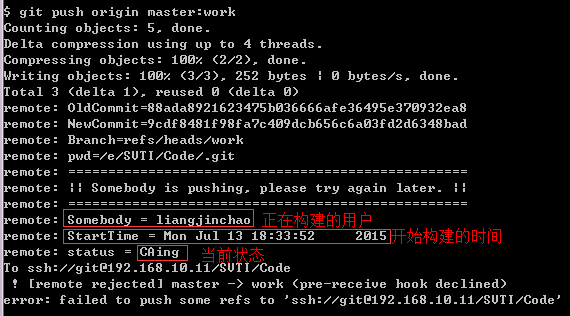


图 3.1 有人正在合入或构建，请等待构建结束

这是最常见的问题。当两个用户同时上传，或者一个用户上传的代码仍在进行测试时，又有新的用户要上传代码时，就会出现这个问题。等待一段时间等持续集成测试完成然后再次上传即可。

遇到这个问题的时候要注意一下**上传者**和**上传时间**是否合理，如图 3.1所示。如果上传者的上传时间（即图中的开始构建的时间）与现在时间相差超过15分钟（持续集成自动化测试的最长时间，具体大小跟编译时间、服务器性能等参数有关），说明出现了系统错误，截图并及时反馈问题给管理员。

## 需要更新代码

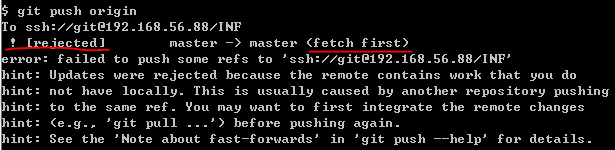


图 3.2 未更新主线代码

上传代码前需要更新代码（fetch）到本地，与本地的修改合并（rebase）才能push到服务器。具体参照2.2.1中的步骤（2）或2.2.2中的步骤（2）。

## 难以消除的CA告警

有时候可能有特殊原因需要及时把代码合入主线，但有的CA告警实在难以消除，可以向管理员申请增加CA告警的数量的限制。但前提是代码要能编译通过而且功能正常。

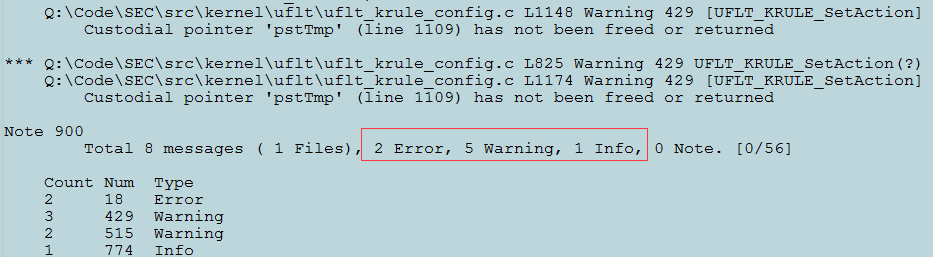


图 3.3 难以消除的CA告警

## 上传分支错误

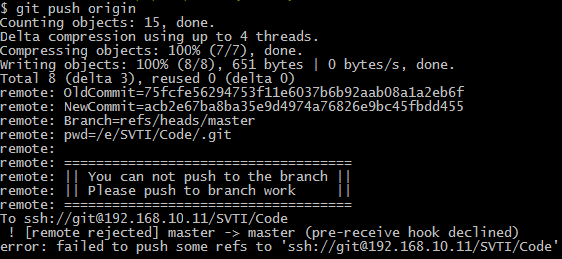


图 3.4 无法上传到该分支

由于持续集成开发环境的需要，用户不能直接上传到到主线分支，而应该上传到work分支。因此应使用命令git push origin master:work进行上传，而不是git push origin。如果是TortoiseGit则参照图 2.8的配置上传。当用户通过自动化测试之后，work分支的内容会自动合入到主线分支。

## 请运行Git\_AutoCA.exe

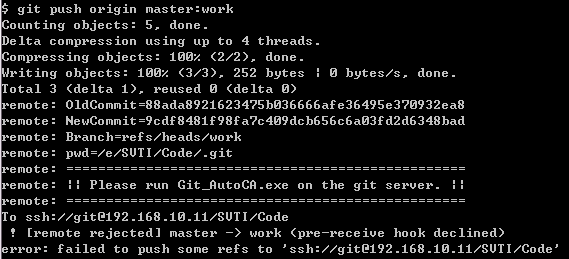


图 3.5 请运行Git\_AutoCA.exe

自动CA程序未启动或遇到致命错误退出。出现这个问题，请及时将情况汇报给管理员，以便及时修复BUG。

## Web页面出现错误调试信息

Jenkins控制台的输出会打印很多程序的调试信息，反应了整个系统的工作状态。当出现一些类似“Error”、“Failed”、“失败”字样的调试信息时，说明系统出现了BUG。将问题及时汇报给管理员以维护持续集成环境。

## 邮件内容有误或乱码

系统错误，及时反馈问题给管理员，并提供当前的构建ID等信息。