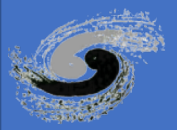
代码管理工具Git

高能物理计算暑期学校

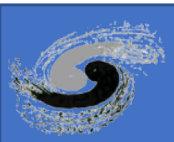
杜然
中国科学院高能物理研究所计算中心

2020-08-24

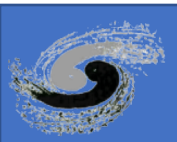


目录

- 什么是代码管理工具？
- 如何安装Git？
- 如何操作Git？
- 多人协作Git工作流？
- 为GitHub上的项目贡献代码？
- 总结一下吧！

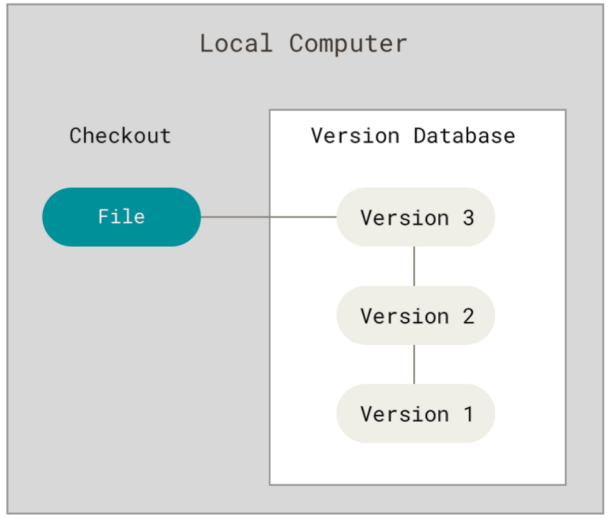


什么是代码管理工具？

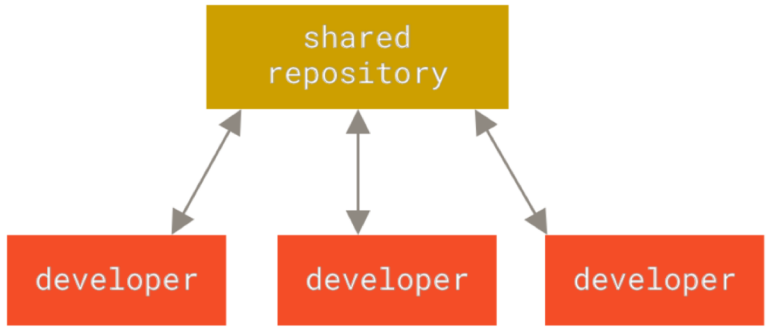


代码管理工具

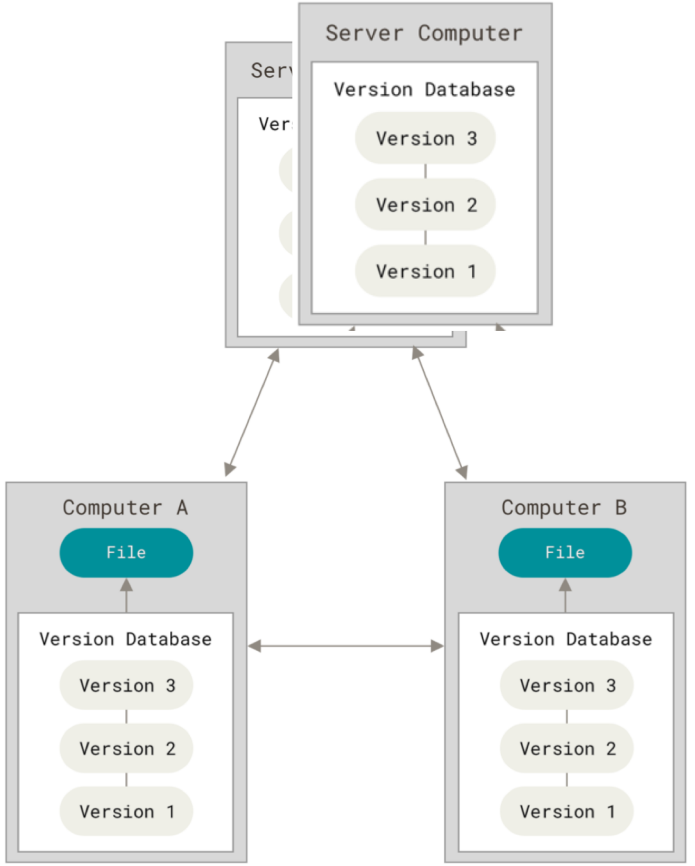
- 功能：记录并管理多个版本的代码文件
- 三种模式
  - 本地：RCS
  - 集中式：CVS, Subversion, Perforce
  - 分布式：Git, Mercurial, Bazaar, Darcs



本地版本控制



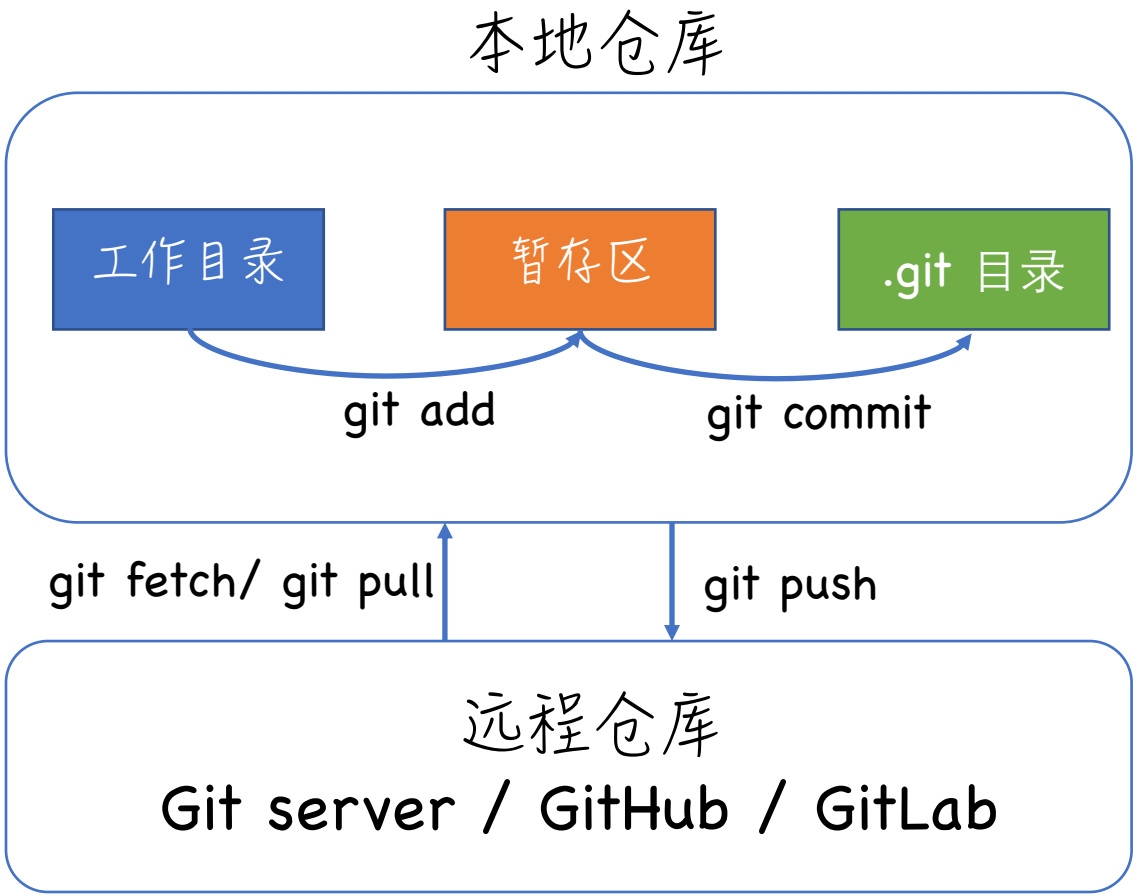
集中式版本控制



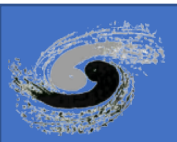
分布式版本控制

Git的一些概念

- 使用Git管理的代码：本地仓库 + 远程仓库
- 几乎所有的操作都可以在本地仓库完成
- 发布代码时，推送到远程仓库
- Git本地仓库的三个区域
  - 工作目录：待操作的文件
  - 暂存区：待提交的文件列表
  - .git仓库目录：元数据+数据库
- Git本地仓库的三种文件状态
  - 已修改
  - 已暂存
  - 已提交



如何安装Git？



安装Git的简单说明

Linux

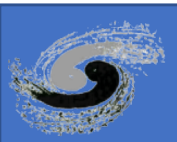
```
# Debian based system : ubuntu
$ sudo apt-get install git-all
# Fedora / CentOS / RHEL
$ sudo dnf install git-all
```

Mac OS

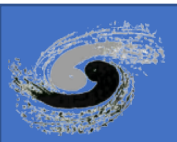
```
# 使用homebrew安装
$ brew install git
# 使用二进制安装文件安装
https://git-scm.com/download/mac
```

Windows

```
# 使用二进制文件安装
https://git-scm.com/download/win
# 如果是win10系统，可使用win10自带的Linux子系统安装Git
```



如何操作Git？



操作前的一些配置

- 配置用户信息

```
$ git config --global user.name "John Smith"
$ git config --global user.email johnsmith@example.com
```

- 配置编辑器

```
$ git config --global core.editor vim
```

- 查看所有配置及所在文件

```
$ git config --list --show-origin
```

- 配置命令：git config
- git使用三级配置文件
  - /etc/gitconfig, --system
  - ~/.gitconfig, --global
  - .git/config, --local
- 自上而下起效

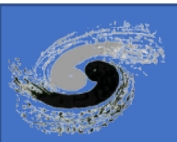
Git操作-准备本地仓库

- 使用已有的目录作为本地仓库

```
$ cd my_project
$ git init
```

- 克隆现有仓库

```
# 克隆本地的另一个目录作为本地仓库
$ git clone /path/to/my_project
# 克隆某台服务器上的目录作为本地仓库
$ git clone username@host:/path/to/my_project
```

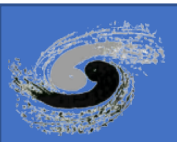


Git操作-工作目录操作

- 根据修改工作目录下的文件
- 查看文件状态

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

```
$ git status -s
M README
MM Rakefile
A lib/git.rb
M lib/simplegit.rb
?? LICENSE.txt
```



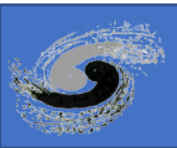
Git操作-暂存区操作

- 暂存已修改的文件

```
$ git add README
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
   (use "git restore --staged <file>..." to unstage)

         new file: README
```

- 使用.gitignore忽略某些文件：密码文件、编译中间文件

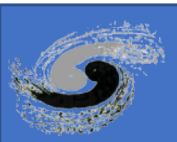


Git操作-.git目录操作

提交更新

```
$ git commit
Add README; Update CONTRIBUTING.MD
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Your branch is up-to-date with 'origin/master'.
#
# Changes to be committed:
#   new file: README
#   modified: CONTRIBUTING.MD
#
~
~
~
README CONTRIBUTING.md
```

```
$ git commit -m 'Add README; Update CONTRIBUTING.MD'
```



Git操作-远程仓库操作

查看远程仓库

```
$ git remote -v
origin https://github.com/schacon/ticgit (fetch)
origin https://github.com/schacon/ticgit (push)
```

本地仓库发布到远程仓库

若使用git clone获取本地仓库

```
$ git push origin master
```

若使用本地非空目录作为本地仓库，并且第一次推送

```
$ git remote add origin <server>
```

Git操作-分支操作

- master是git clone时系统默认的分支，可用于保存代码稳定版
- 分支可用于实现某单一功能，Debug等。分支代码稳定后，可并入master
- 创建分支

```
$ git branch <branch_name>
$ git checkout <branch_name>
# 等于上面两条命令
$ git checkout -b <branch_name>
```

- 合并分支

```
$ git checkout master
$ git merge <branch_name>
$ git branch -d <branch_name>
```

Git操作-标签操作

- 标签可用于发布代码稳定版本，2种
  - 轻量标签：某个特定提交的引用
  - 附注标签：推荐，包含操作信息完整，可
- 创建标签

```
$ git tag -a <tag_name> -m "<some_comments>"
```

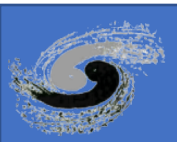
- 查看标签

```
$ git tag -l
```

- 共享标签

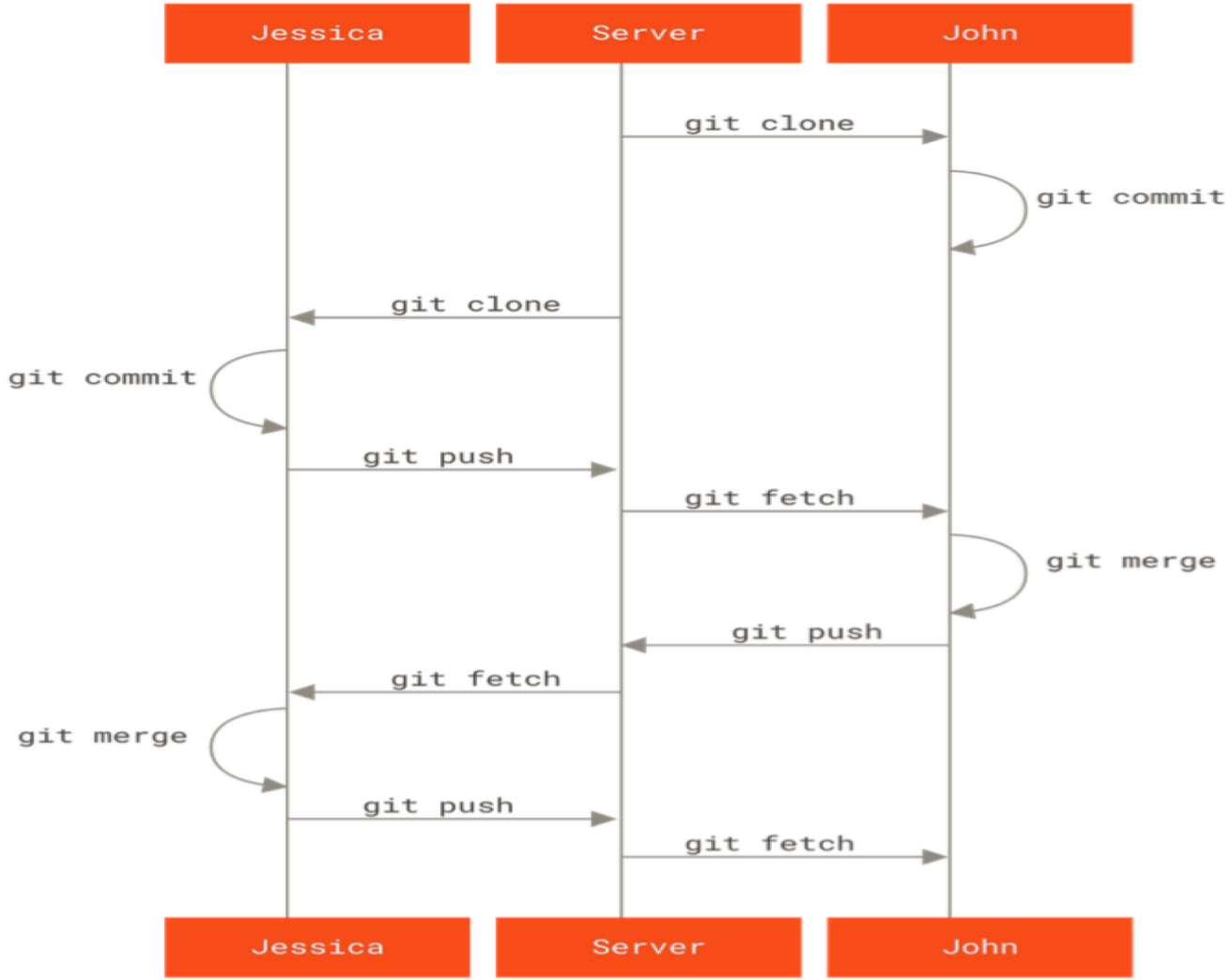
```
$ git push <remote> <tag_name>
```

多人协作Git工作流？



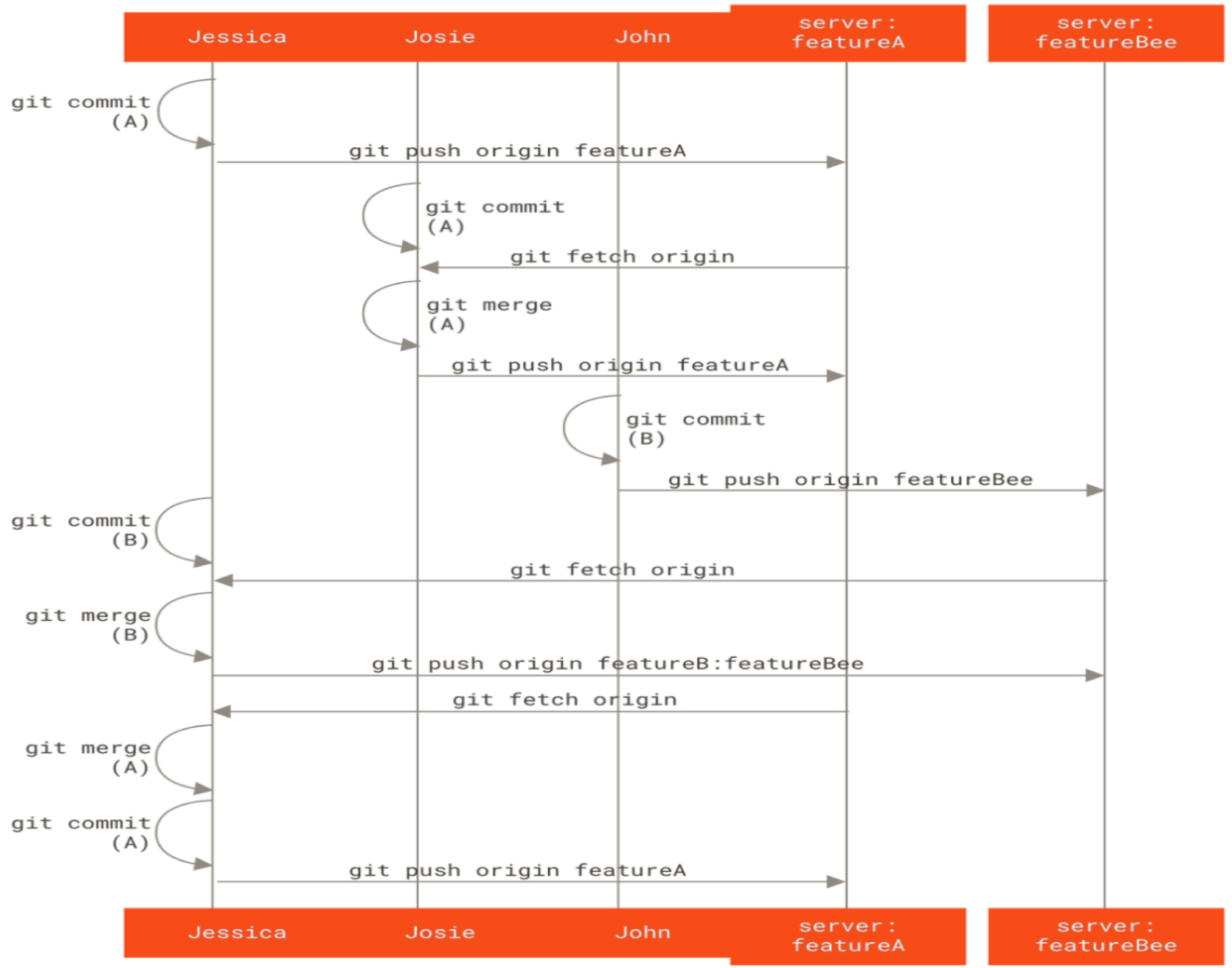
私有小型团队的Git工作流

- 非公开项目
- 项目较小型
- 开发者均有远程仓库master分支的推送权限

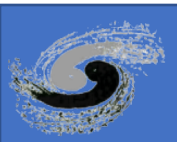


私有管理团队的工作流

- 非公开项目，项目较大型
- 开发人员分为小组，每个小组专注一个功能
- 开发人员的代码更新由管理者审核
- 只有管理者可将审核后的代码合并到master分支

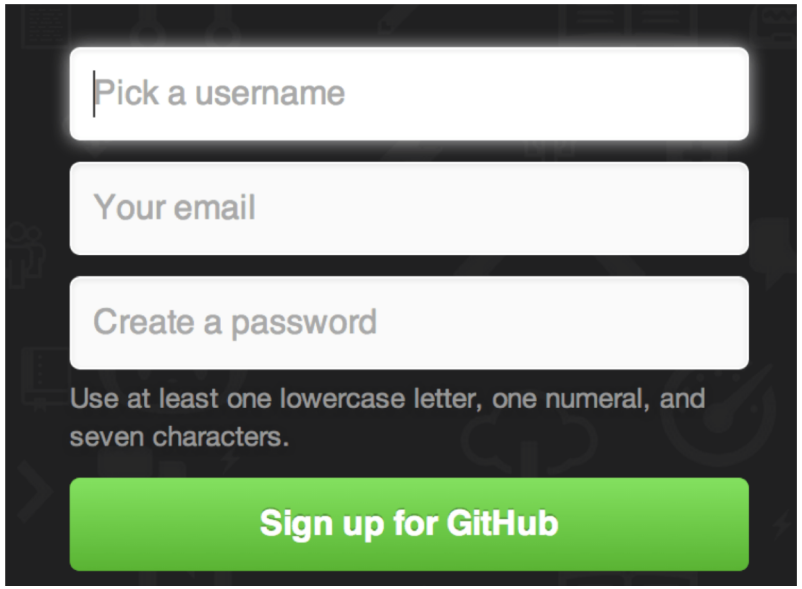


为GitHub上的项目贡献代码？

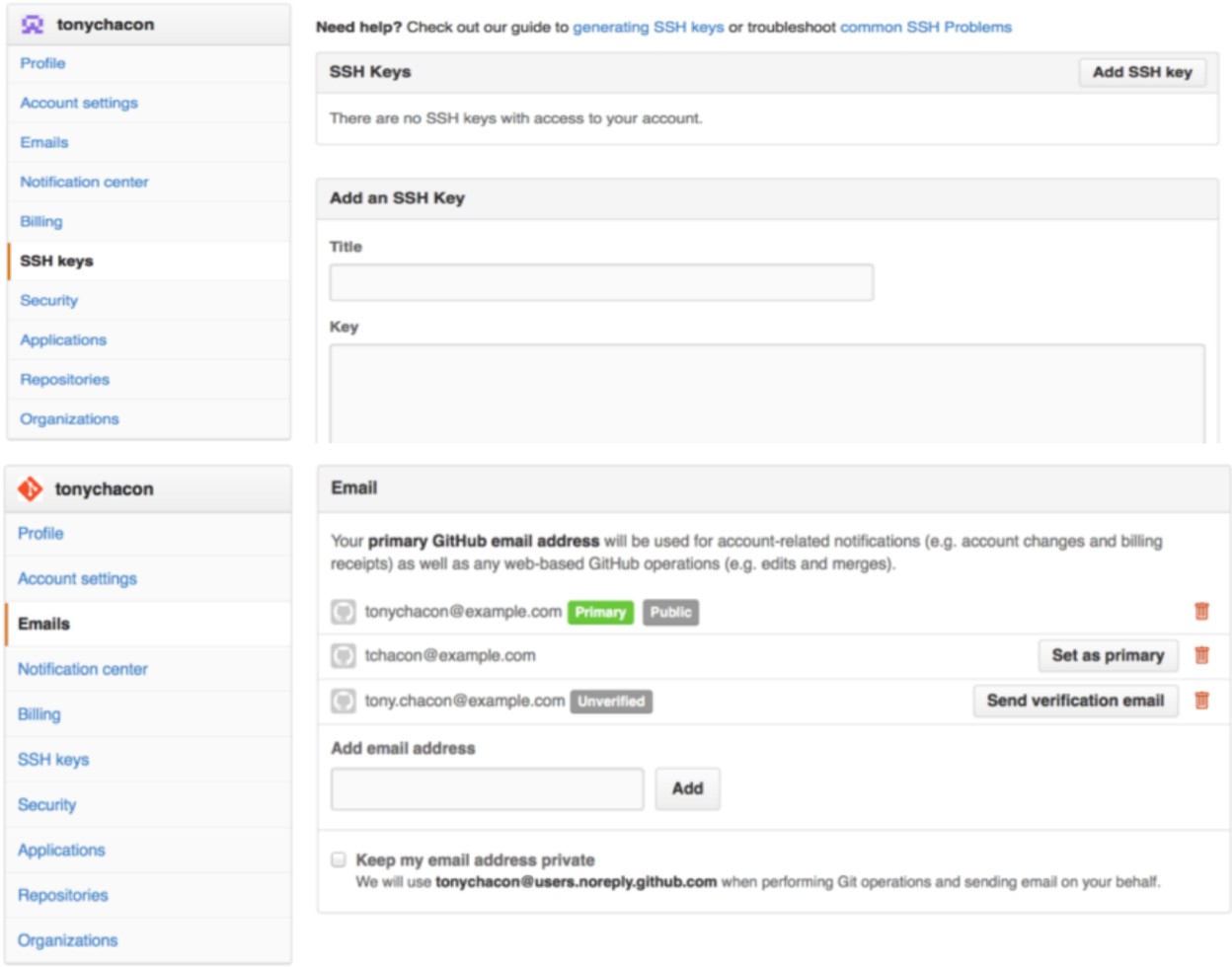


GitHub账号创建及配置

- 创建账号、上传SSH keys、添加邮件地址



创建账号

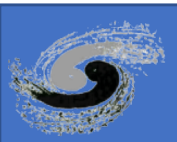


上传
SSH keys

添加
邮件地址

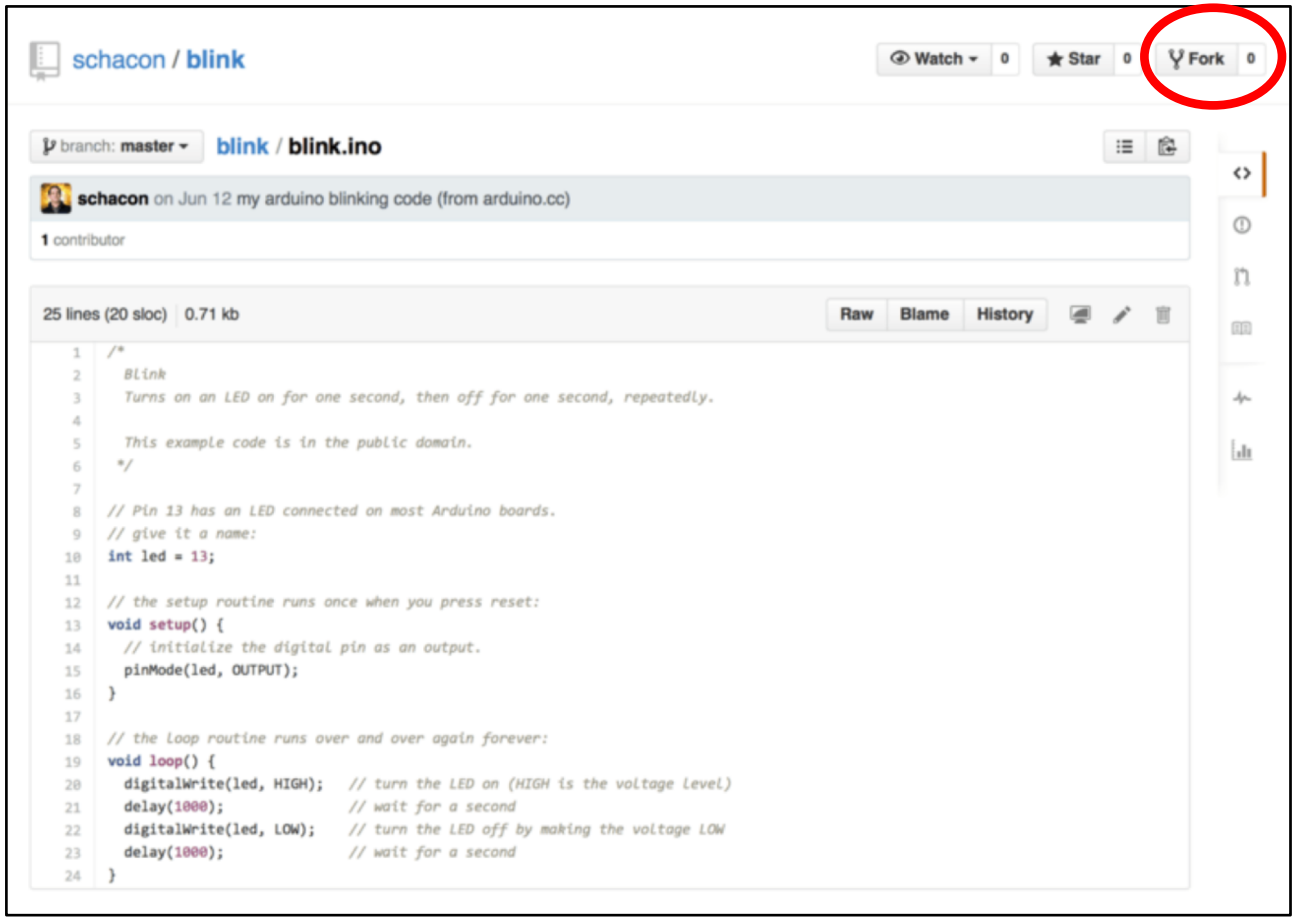
贡献代码流程

- 自原生项目派生（fork）一个项目 ← GitHub操作
- 克隆（clone）派生项目到本地仓库
- 在本地仓库中创建一个新的分支（branch）
- 在分支上修改代码（add，commit）
- 将修改好的代码推送（push）到GitHub上的派生仓库 ← 命令行操作
- 创建一个拉取请求（pull-request）
- 讨论/根据实际情况修改代码 ← GitHub操作
- 原生项目管理者合并/关闭拉取请求
- 将更新后的原生项目master分支同步到派生项目 ← 命令行操作

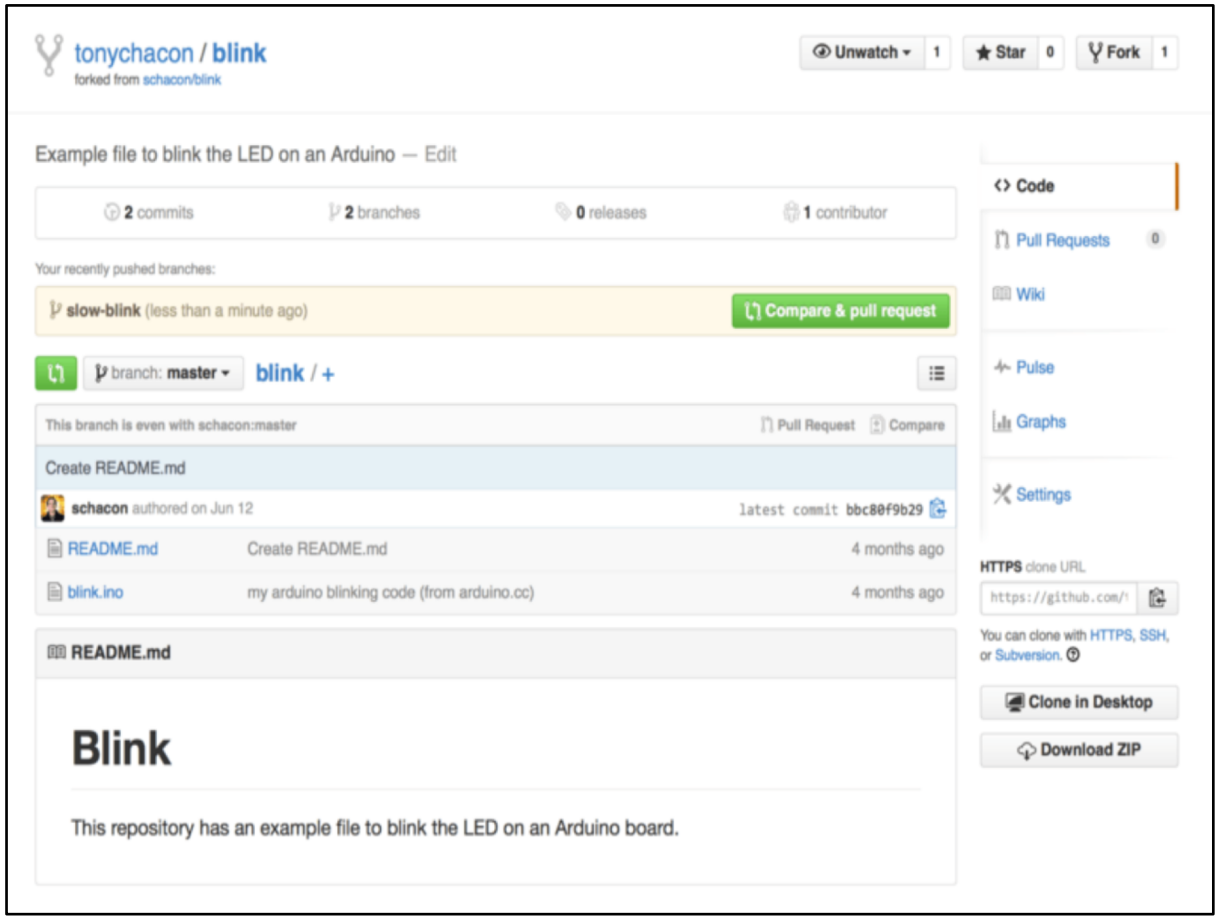


派生

原生项目


schacon / blink
Watch 0
Star 0
Fork 0
branch: master
blink / blink.ino
schacon on Jun 12 my arduino blinking code (from arduino.cc)
1 contributor
25 lines (20 sloc) 0.71 kb
Raw Blame History
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.
This example code is in the public domain.
*/
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pin as an output.
pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000); // wait for a second
digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
delay(1000); // wait for a second
}

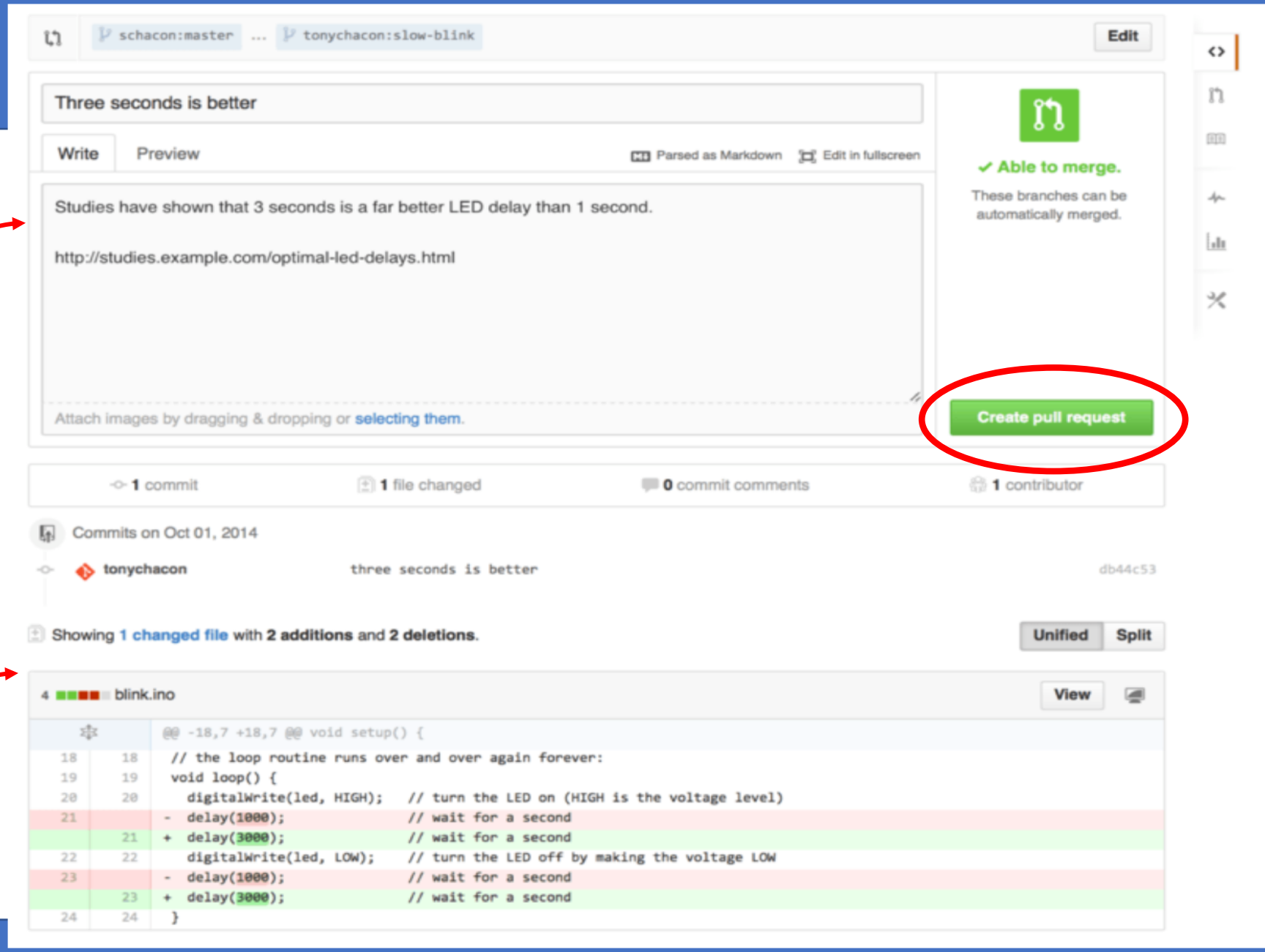
派生项目


tonychacon / blink
forked from schacon/blink
Unwatch 1
Star 0
Fork 1
Example file to blink the LED on an Arduino — Edit
2 commits
2 branches
0 releases
1 contributor
Your recently pushed branches:
slow-blink (less than a minute ago)
Compare & pull request
branch: master
blink / +
This branch is even with schacon:master
Pull Request
Compare
Create README.md
schacon authored on Jun 12
latest commit bbc80f9b29
README.md Create README.md 4 months ago
blink.ino my arduino blinking code (from arduino.cc) 4 months ago
README.md
Blink
This repository has an example file to blink the LED on an Arduino board.
Code
Pull Requests 0
Wiki
Pulse
Graphs
Settings
HTTPS clone URL
https://github.com/t
You can clone with HTTPS, SSH, or Subversion.
Clone in Desktop
Download ZIP

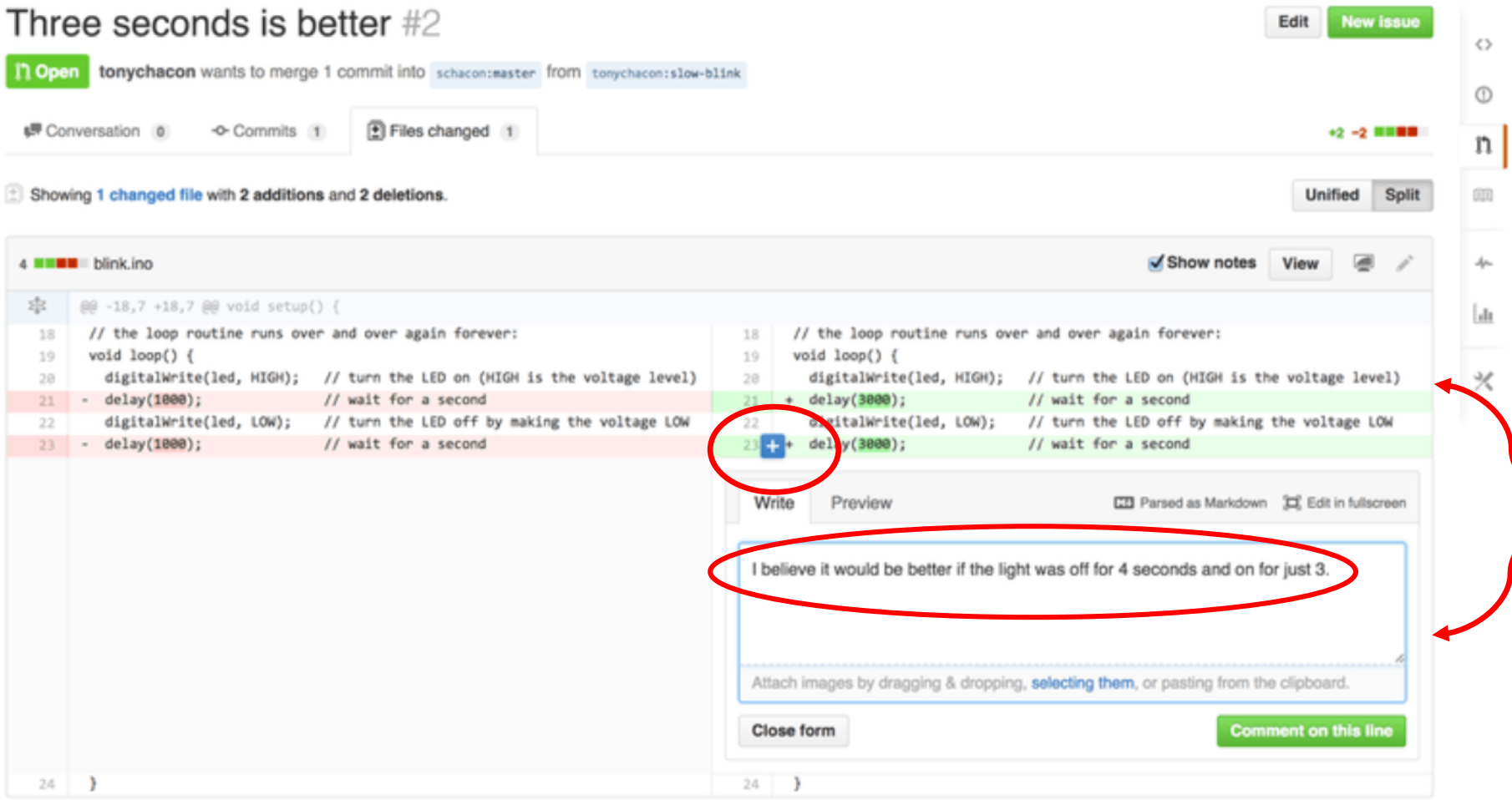
拉取请求

写清楚代码
更改缘由

与原生代码
的比较

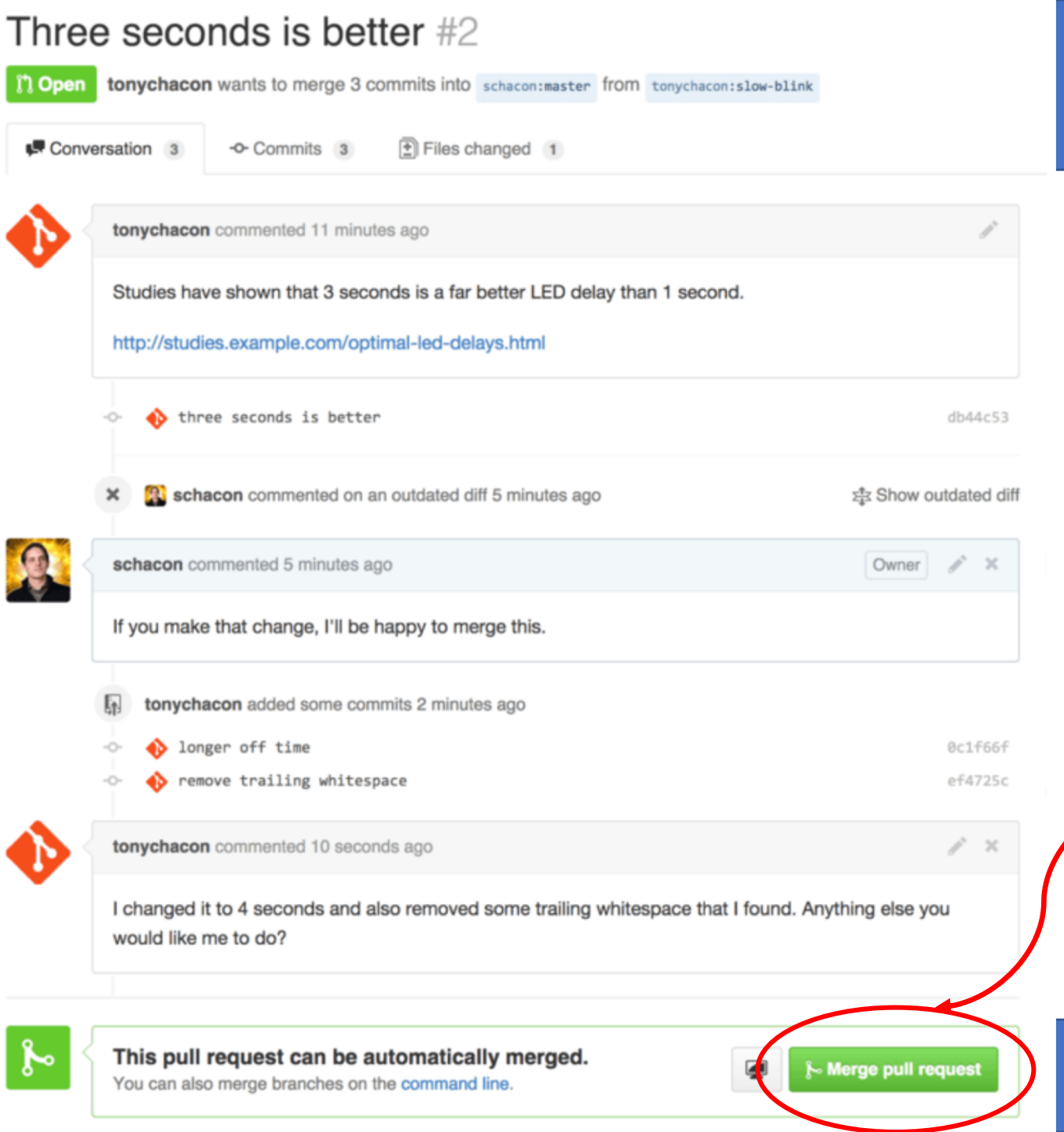


讨论修改



原生项目管理者
可在每处更改
增加评论

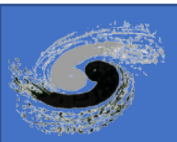
合并请求



讨论和修改
全过程

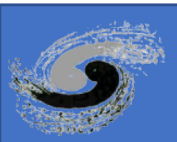
合并

总结一下吧！



我们都学到了什么？

- 代码管理工具的种类
- Git管理代码的工作方式
- 安装Git工具
- Git的基本操作命令
- 多人协作的Git工作流
- GitHub贡献代码的方法

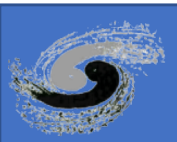


就到这里。。。

休息。。。休息一下吧！

Git操作小结

- 准备代码仓库
  - git init
  - git clone
- 操作配置文件
  - git config
- 代码文件操作
  - git status
  - git diff
  - git rm
  - git mv
- 暂存区操作
  - git add
  - git commit
  - git log
- 远程仓库操作
  - git remote add/show/rename/remove
  - git fetch
  - git pull
  - git push
- 标签操作
  - git tag -l
  - Git tag -a <tag_name> -m "some comments"
  - git show <tag_name>
  - Git push <remote> <tag_name>
  - Git push <remote> --tags
  - Git tag -d <tag_name>
  - Git push <remote> --delete <tag_name>
- 别名操作（可选）
  - Git config --global alias.<alias_cmd> "<cmd>"
- 分支操作
  - git branch <branch>
  - git checkout <branch>
  - git log --oneline --decorate --graph -all
  - Git merge <branch>
  - Git branch -d <branch>
  - git checkout --track <remote>/<branch>
  - git branch -vv
  - Git push origin --delete <branch>
- 变基操作（可选）
  - Git rebase <base_branch> <topic_branch>

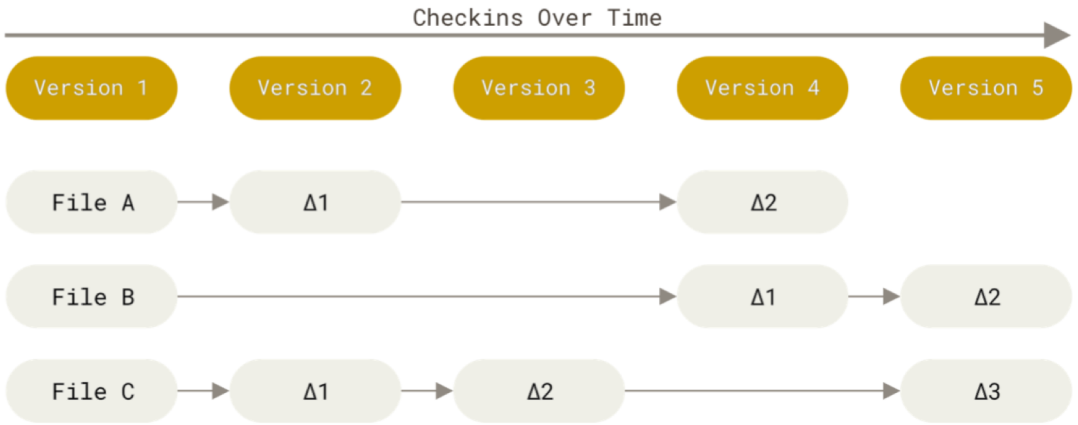


参考资料

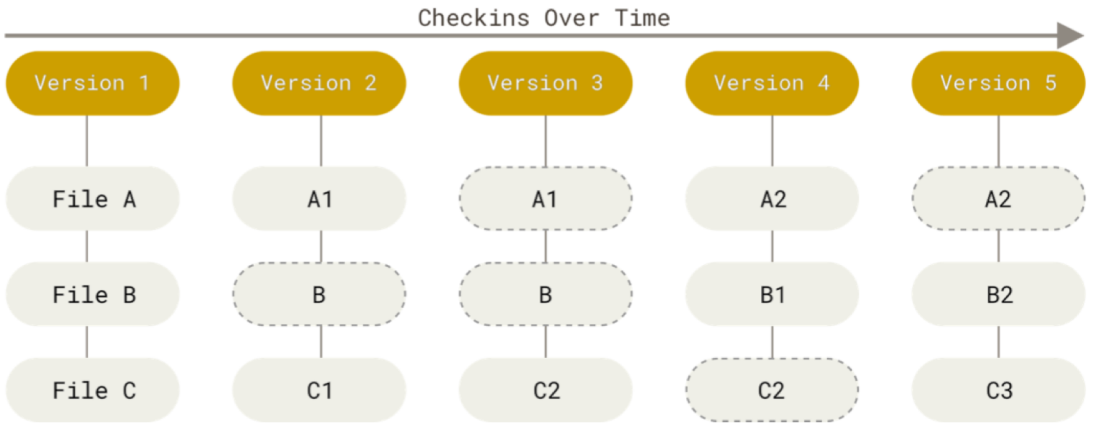
- 还想学习更多Git功能，就看看这些吧！
  - Git入门
    - http://rogerdudler.github.io/git-guide/index.zh.html
  - Git Pro 2 ： 本次报告参考的主要书籍
    - https://git-scm.com/book/zh/v2
  - GitHub help
    - https://docs.github.com/en
  - 图解Git
    - http://marklodato.github.io/visual-git-guide/index-zh-cn.html
  - Think Like a Git
    - http://think-like-a-git.net/
  - Git社区
    - https://book.git-scm.com/

Git保存文件快照

- Git记录文件快照，而非差异
- 采用SHA-1校验文件：数据完整性 + 标识唯一性



记录文件与初始版本的差异



直接存储每个版本的文件快照