



# git

## git & gitlab

李云@天阙科技

# 目录

- Git简介
- Git常用命令&概念
- SSH认证流程
- Gitflow
- Gitlab & 项目管理 & 文档体系
- pull request 代码审阅
- .gitignore

# git简介

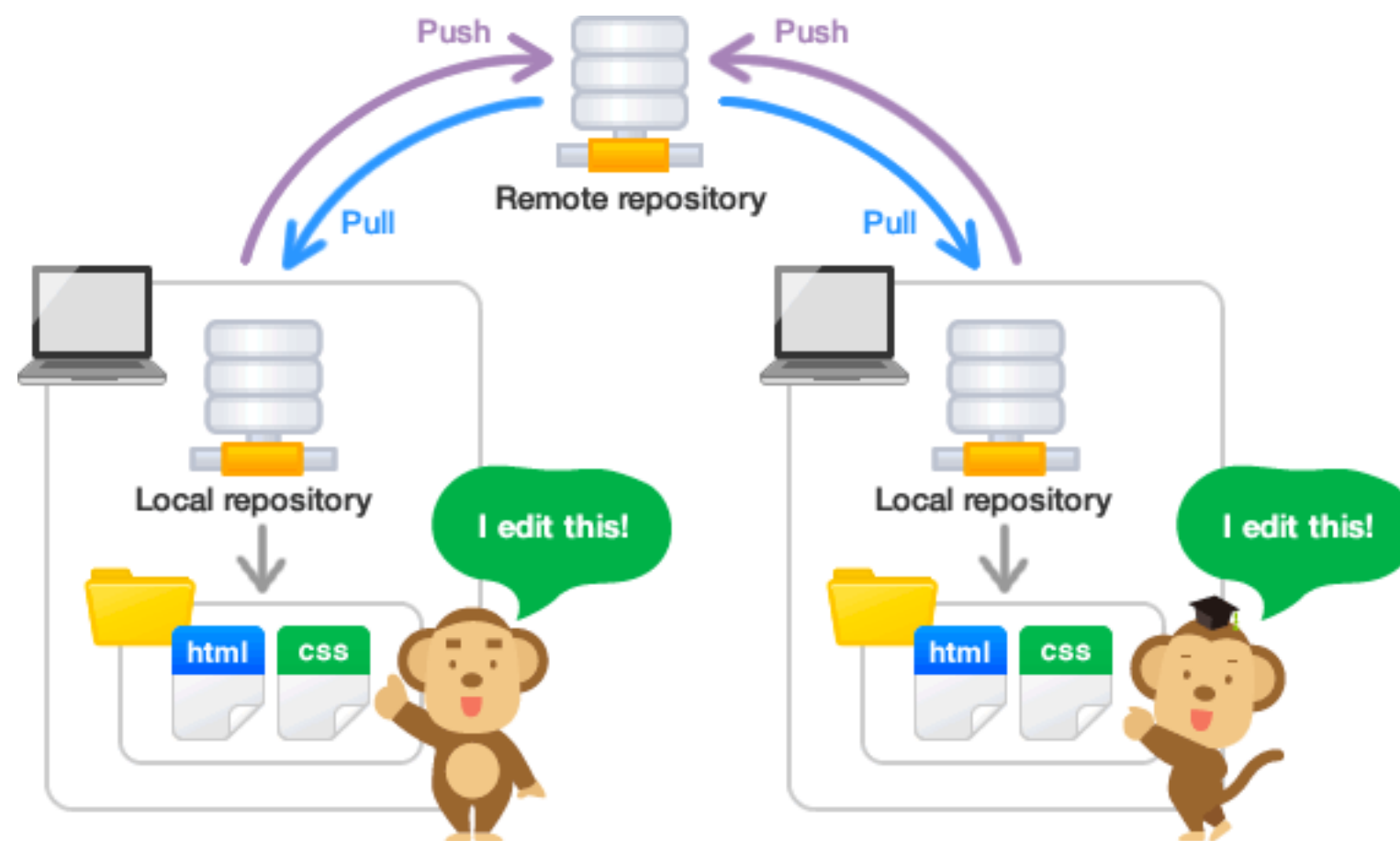
Git是一个分布式版本控制系统（这类系统还包括 Mercurial, Bazaar 以及 Darcs 等），原是Linux内核開發者林纳斯·托瓦兹（Linus Torvalds）为更好地管理Linux内核开发而设计。

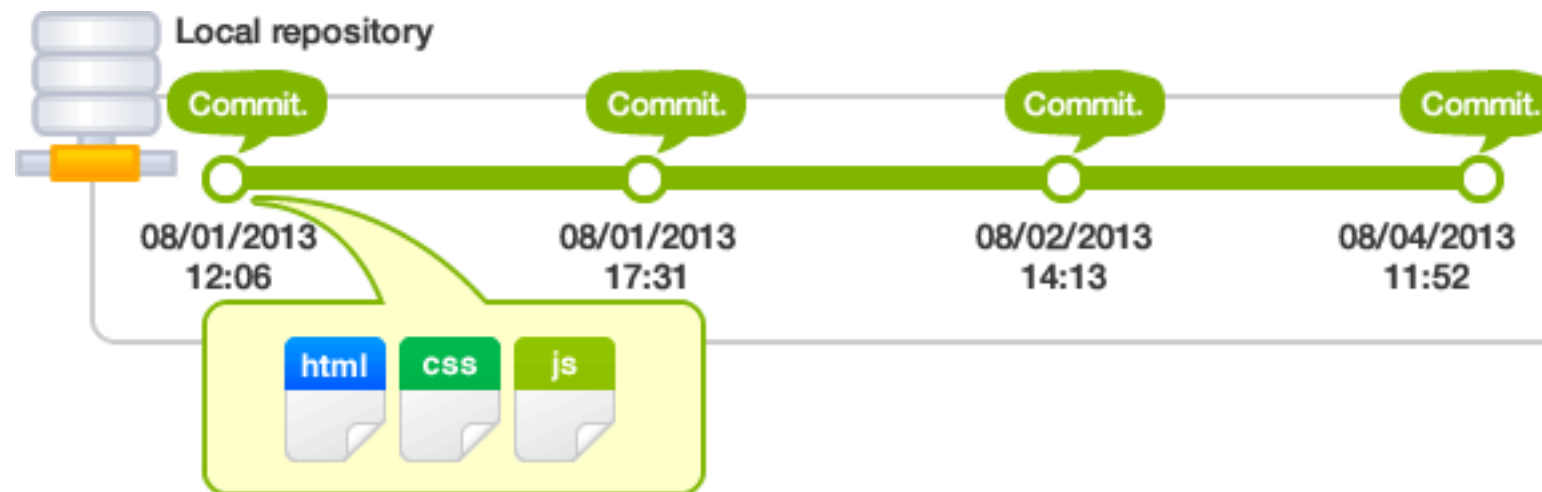
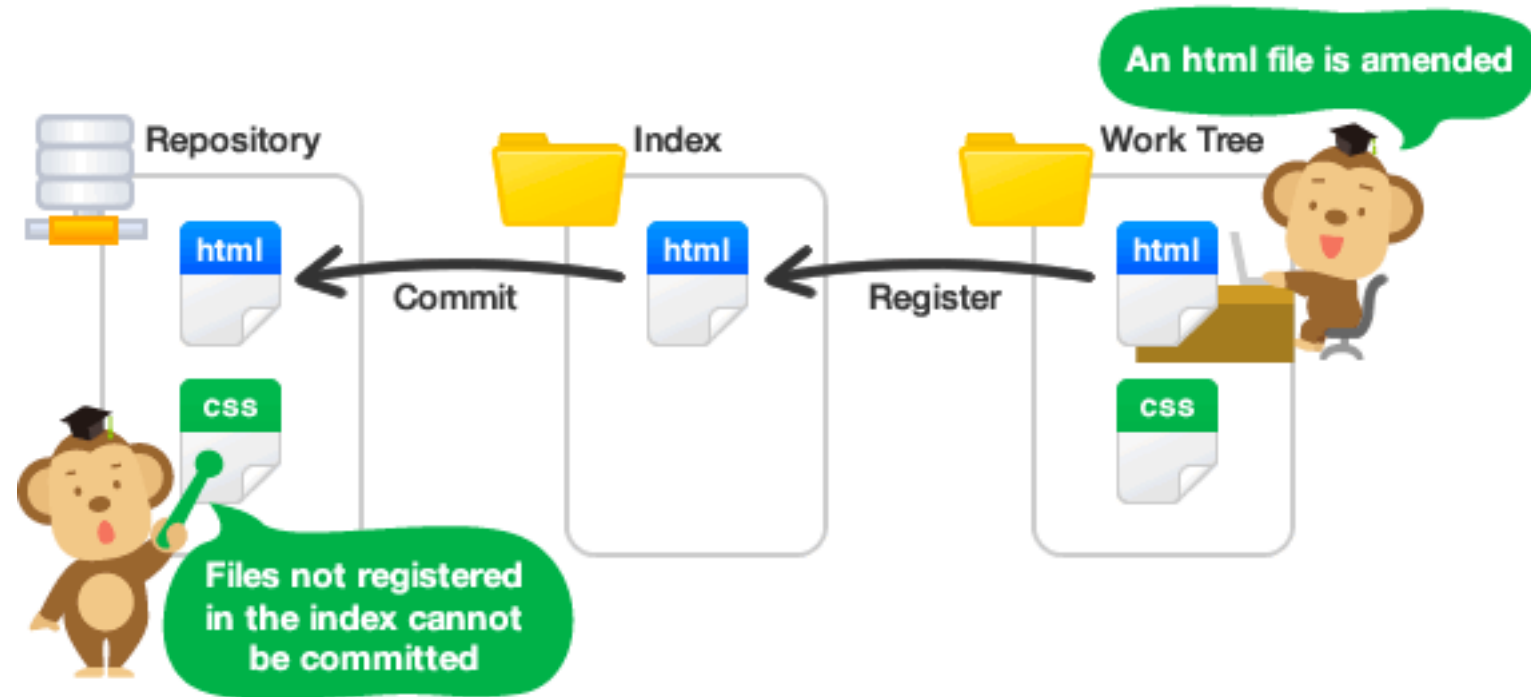
Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

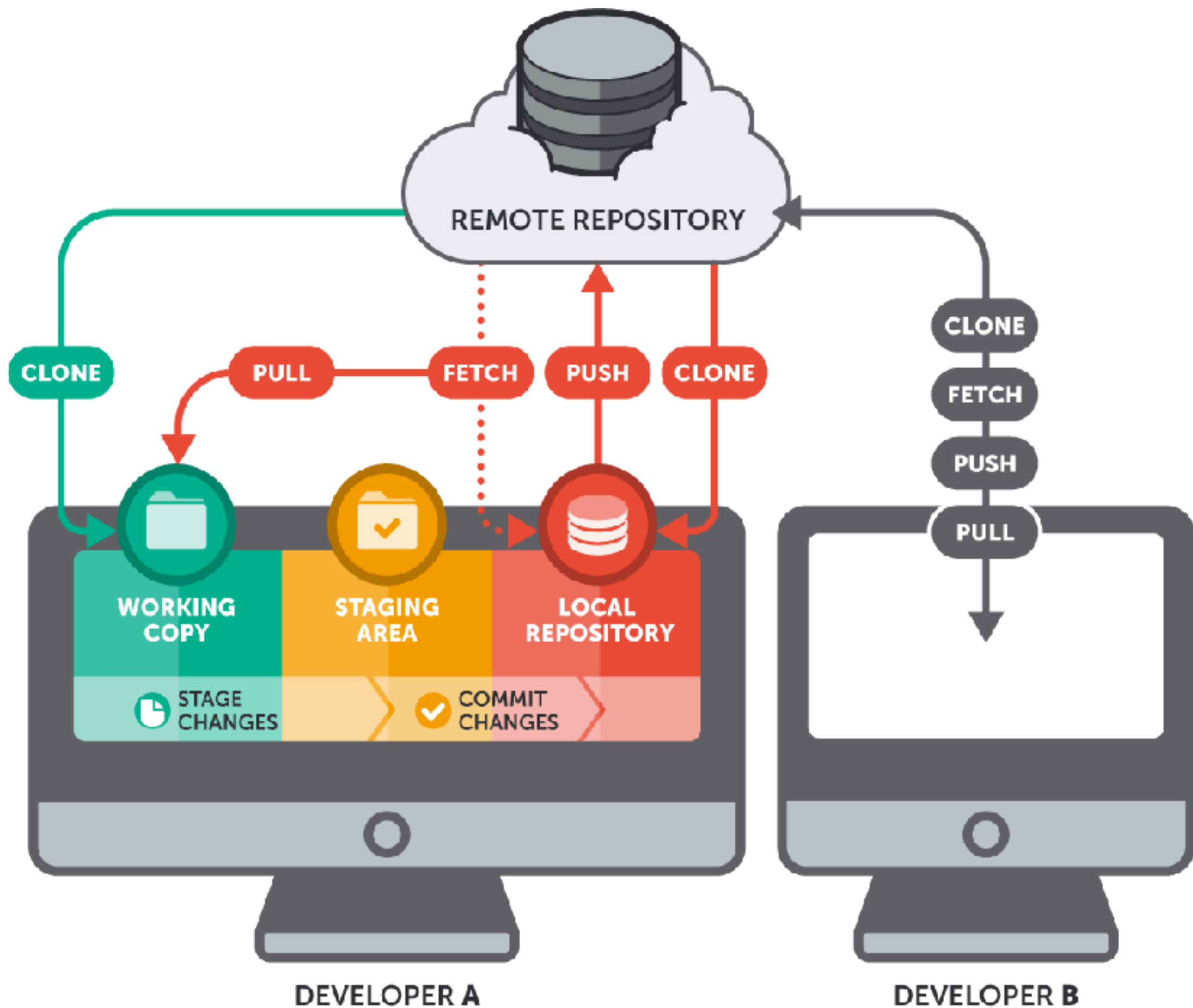
Git is **easy to learn** and has a tiny footprint with **lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

# git的优点

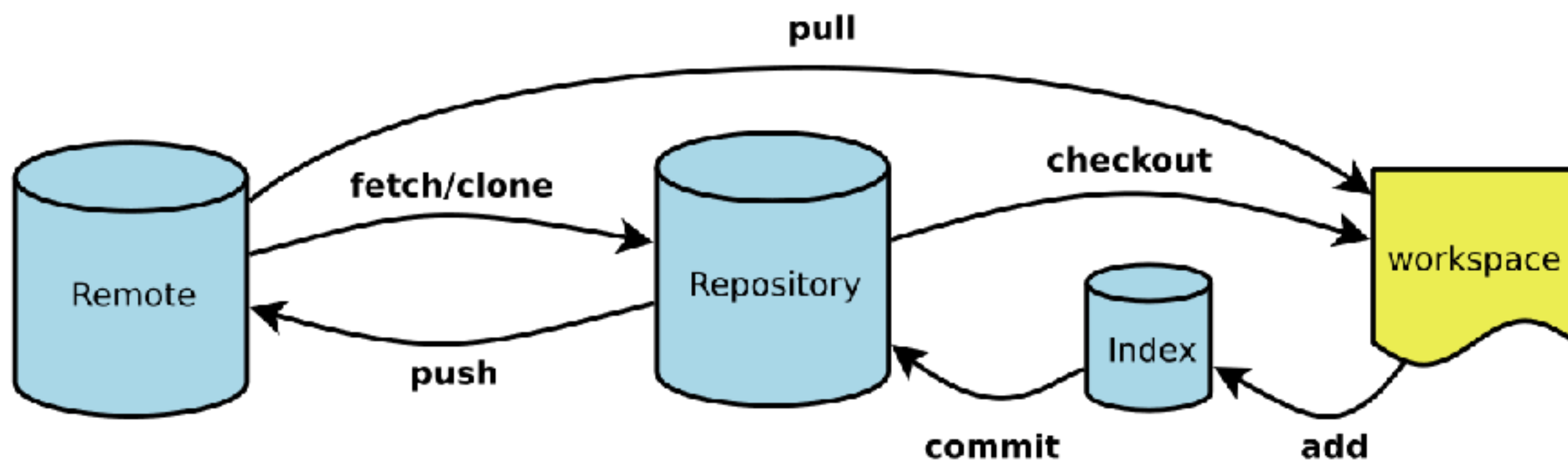
- 便宜的本地分支
- 所有内容都在本地
- Git 很快
- Git 很小巧
- 暂存区域
- 分布式
- 适用于任何工作流







# git常用命令





Try Git

# Git 常用命令速查表

master :默认开发分支  
origin :默认远程版本库

Head :默认开发分支  
Head^ :Head 的父提交

## 创建版本库

```
$ git clone <url>          #克隆远程版本库
$ git init                 #初始化本地版本库
```

## 修改和提交

```
$ git status              #查看状态
$ git diff                #查看变更内容
$ git add .               #跟踪所有改动过的文件
$ git add <file>          #跟踪指定的文件
$ git mv <old> <new>      #文件改名
$ git rm <file>           #删除文件
$ git rm --cached <file>  #停止跟踪文件但不删除
$ git commit -m "commit message"
                           #提交所有更新过的文件
$ git commit --amend      #修改最后一次提交
```

## 查看提交历史

```
$ git log                 #查看提交历史
$ git log -p <file>       #查看指定文件的提交历史
$ git blame <file>        #以列表方式查看指定文件的提交历史
```

## 撤消

```
$ git reset --hard HEAD  #撤消工作目录中所有未提交文件的修改内容
$ git checkout HEAD <file> #撤消指定的未提交文件的修改内容
$ git revert <commit>    #撤消指定的提交
```

## 分支与标签

```
$ git branch              #显示所有本地分支
$ git checkout <branch/tag> #切换到指定分支或标签
$ git branch <new-branch> #创建新分支
$ git branch -d <branch>  #删除本地分支
$ git tag                 #列出所有本地标签
$ git tag <tagname>       #基于最新提交创建标签
$ git tag -d <tagname>    #删除标签
```

## 合并与衍合

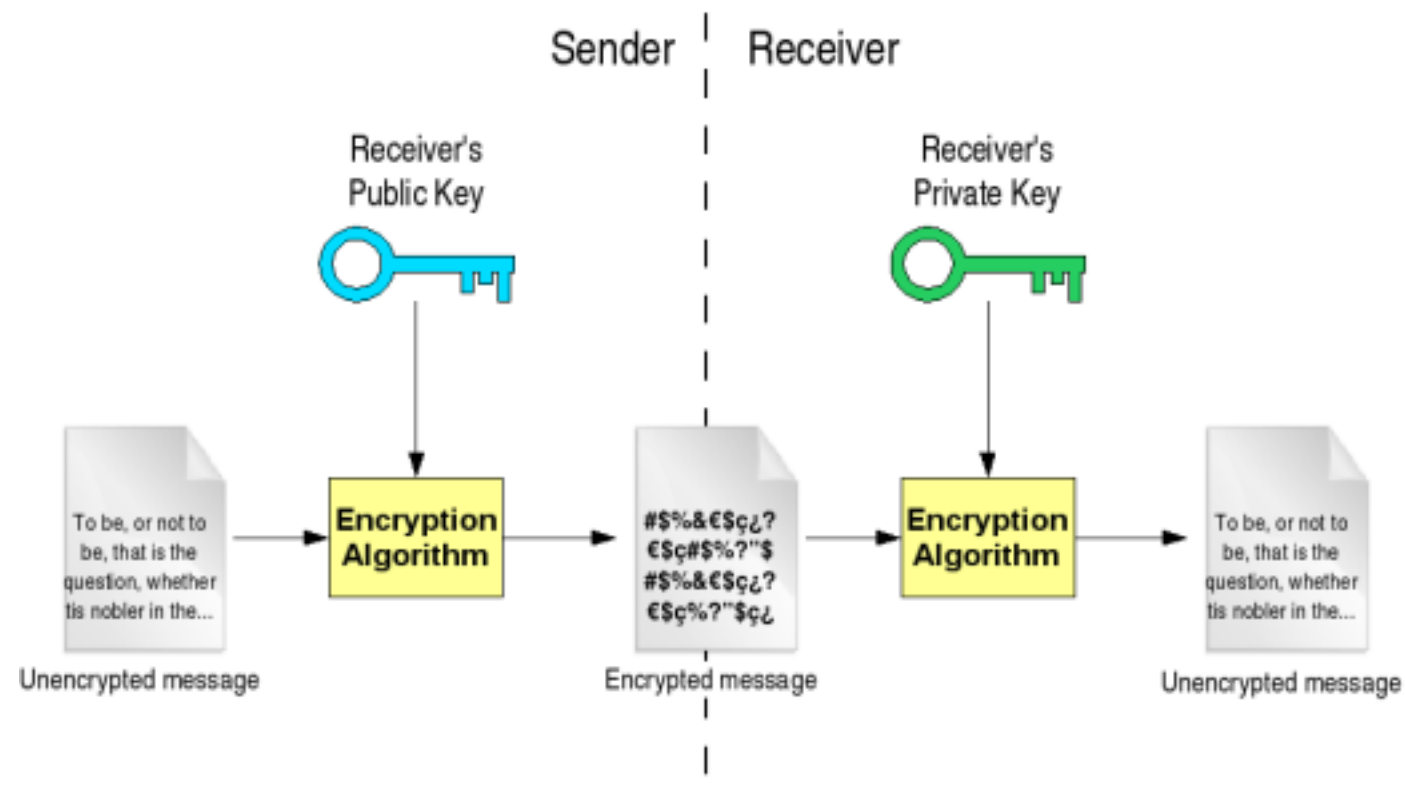
```
$ git merge <branch>      #合并指定分支到当前分支
$ git rebase <branch>     #衍合指定分支到当前分支
```

## 远程操作

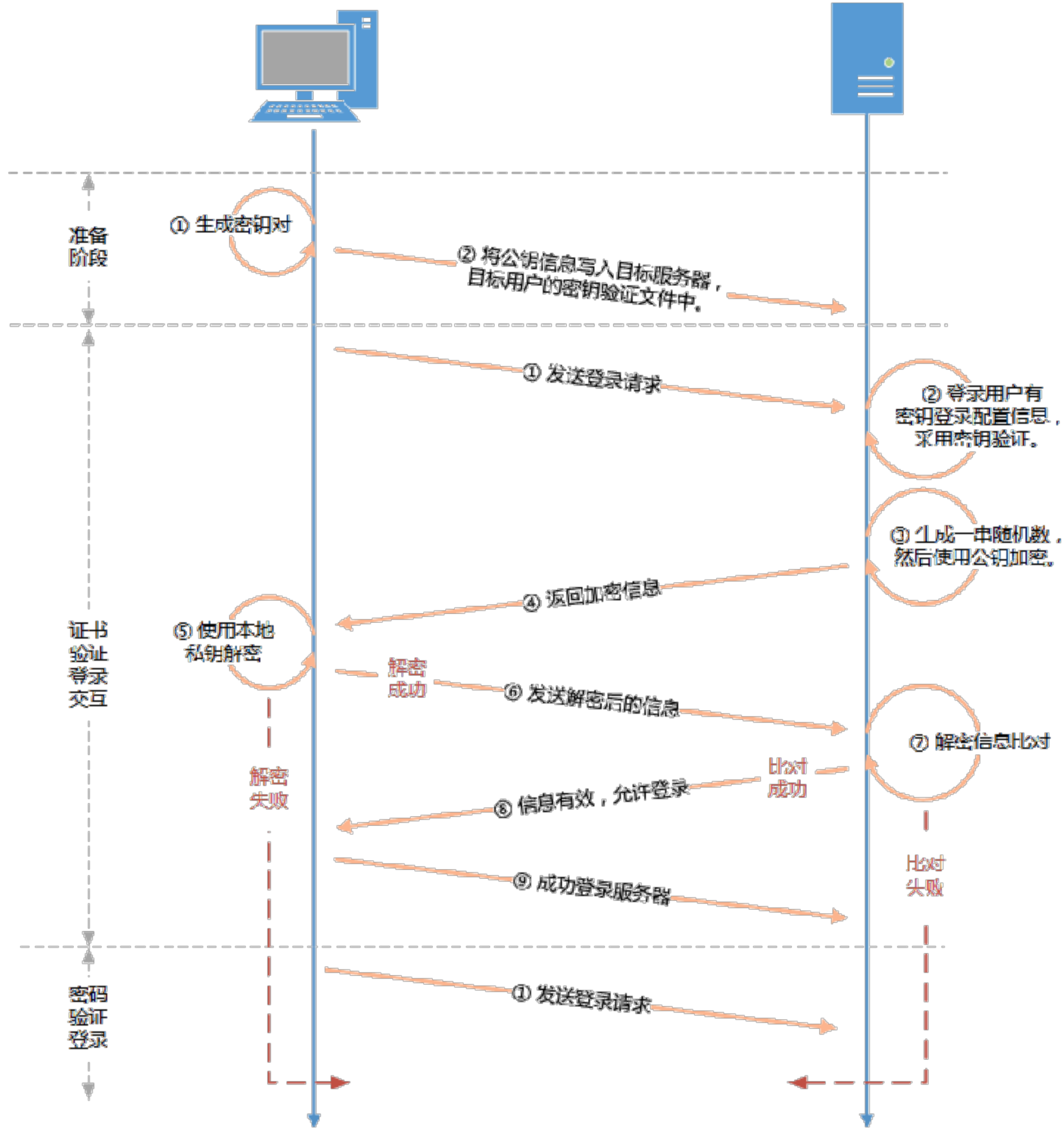
```
$ git remote -v           #查看远程版本库信息
$ git remote show <remote> #查看指定远程版本库信息
$ git remote add <remote> <url>
                           #添加远程版本库
$ git fetch <remote>      #从远程库获取代码
$ git pull <remote> <branch> #下载代码及快速合并
$ git push <remote> <branch> #上传代码及快速合并
$ git push <remote> :<branch/tag-name>
                           #删除远程分支或标签
$ git push --tags         #上传所有标签
```

# 公钥认证

Ac 客户端公钥  
Bc 客户端密钥  
As 服务器公钥  
Bs 服务器密钥



```
ssh-keygen -t rsa -C 'email'
```

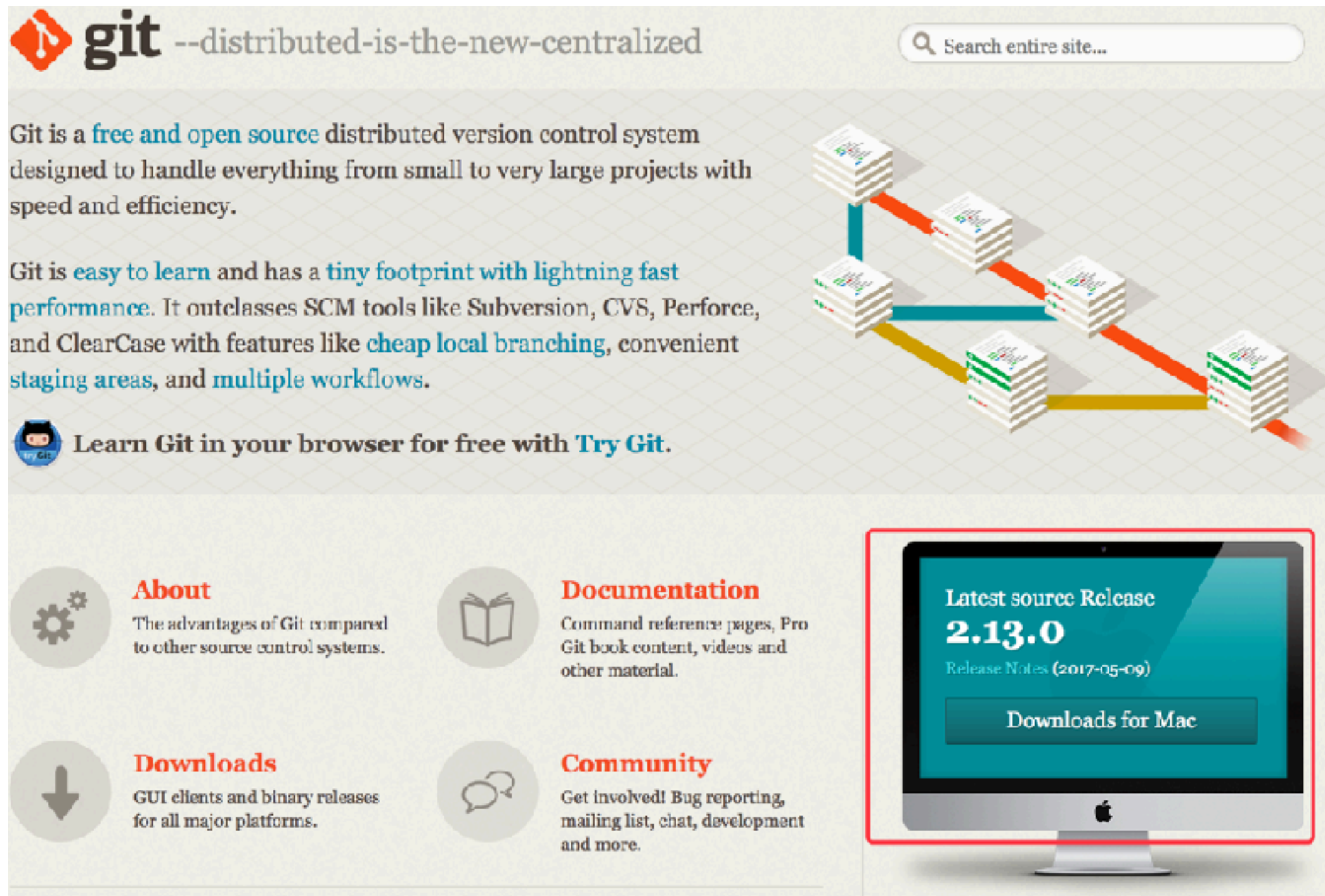


- 会话密钥(session key)生成
  - 客户端请求连接服务器，服务器将  $A_s$  发送给客户端。
  - 服务器生成会话ID(session id)，设为  $p$ ，发送给客户端。
  - 客户端生成会话密钥(session key)，设为  $q$ ，并计算  $r = p \text{ xor } q$ 。
  - 客户端将  $r$  用  $A_s$  进行加密，结果发送给服务器。
  - 服务器用  $B_s$  进行解密，获得  $r$ 。
  - 服务器进行  $r \text{ xor } p$  的运算，获得  $q$ 。
  - 至此服务器和客户端都知道了会话密钥 $q$ ，以后的传输都将被  $q$  加密。

- 认证
  - 服务器生成随机数  $x$ ，并用  $A_c$  加密后生成结果  $S(x)$ ，发送给客户端
  - 客户端使用  $B_c$  解密  $S(x)$  得到  $x$
  - 客户端计算  $q + x$  的 md5 值  $n(q+x)$ ， $q$ 为上一步得到的会话密钥
  - 服务器计算  $q + x$  的 md5 值  $m(q+x)$
  - 客户端将  $n(q+x)$  发送给服务器
  - 服务器比较  $m(q+x)$  和  $n(q+x)$ ，两者相同则认证成功



# 安装git




The image shows the Git website banner. At the top left is the Git logo (an orange diamond with a white branching diagram) followed by the text "git --distributed-is-the-new-centralized". To the right is a search bar with the placeholder text "Search entire site...". Below the header, there are two paragraphs of text. The first paragraph describes Git as a free and open source distributed version control system. The second paragraph describes Git as easy to learn and having a tiny footprint with lightning fast performance. To the right of the text is a diagram showing several stacks of papers connected by lines, representing a distributed version control system. Below the text, there is a "Try Git" button with a GitHub logo. At the bottom, there are four sections: "About", "Documentation", "Downloads", and "Community". Each section has an icon and a brief description. To the right of these sections is a large monitor displaying the "Latest source Release 2.13.0" and a "Downloads for Mac" button. The monitor is highlighted with a red border.

**git** --distributed-is-the-new-centralized

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

 Learn Git in your browser for free with **Try Git**.

**About**  
The advantages of Git compared to other source control systems.

**Documentation**  
Command reference pages, Pro Git book content, videos and other material.

**Downloads**  
GUI clients and binary releases for all major platforms.

**Community**  
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release  
**2.13.0**  
Release Notes (2017-05-09)  
[Downloads for Mac](#)

# 生成ssh key

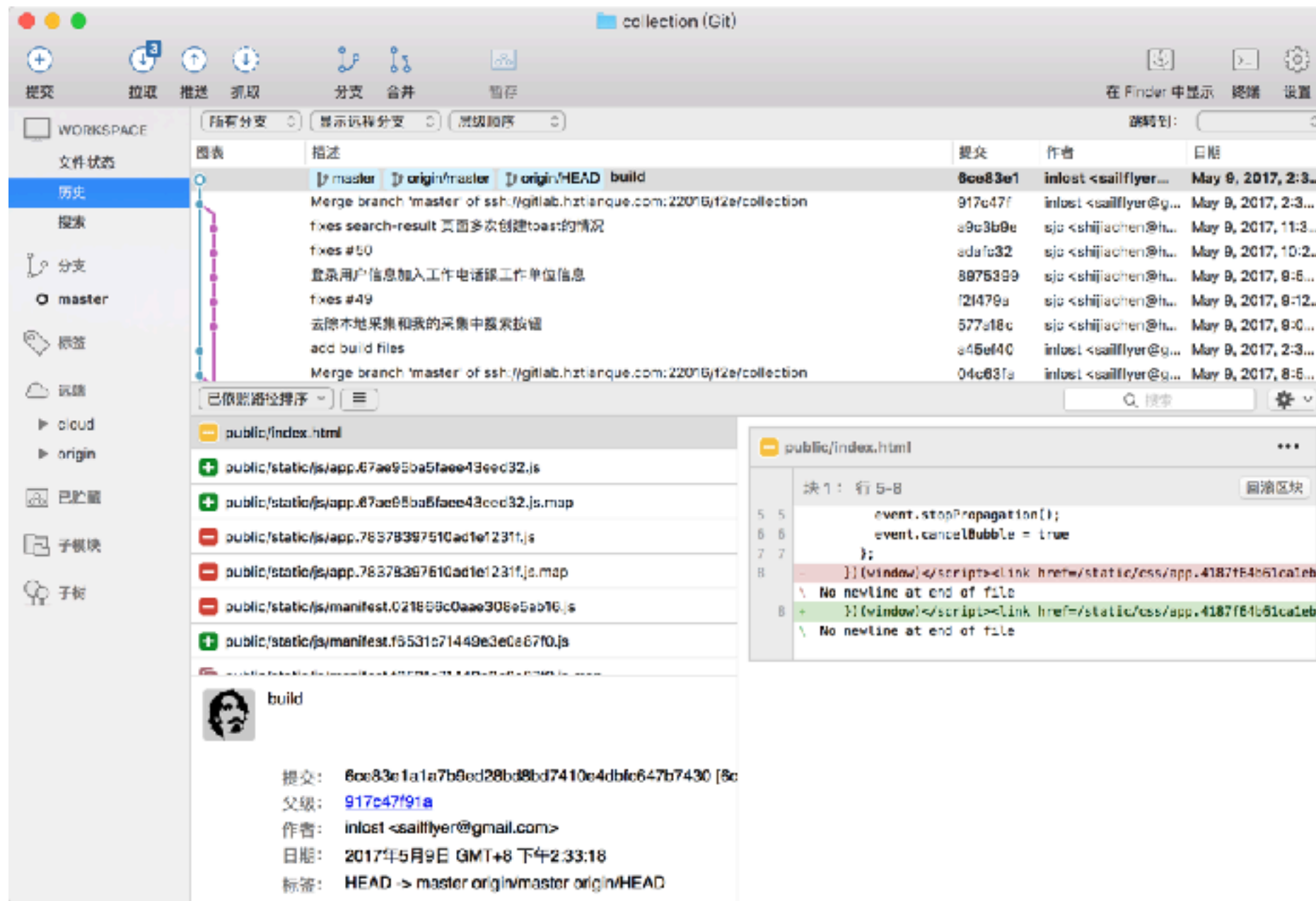
- windows打开 git bash, linux,mac打开命令行
- 在git bash 下输入 `ssh-keygen -t rsa -C "$Your e-mail"`
- 按三下回车键
- 就在可以在当前用户的的家目录下看到.ssh文件夹
- `id_rsa.pub`是公钥, `id_rsa`是私钥



# 添加key到git server

- 打开[gitlab.hztianque.com](https://gitlab.hztianque.com)
- 登录后点击右上角的头像选择settings
- 选择ssh keys
- 将刚刚生成的id\_rsa.pub的内容粘贴到key的文本框
- 点击add key保存

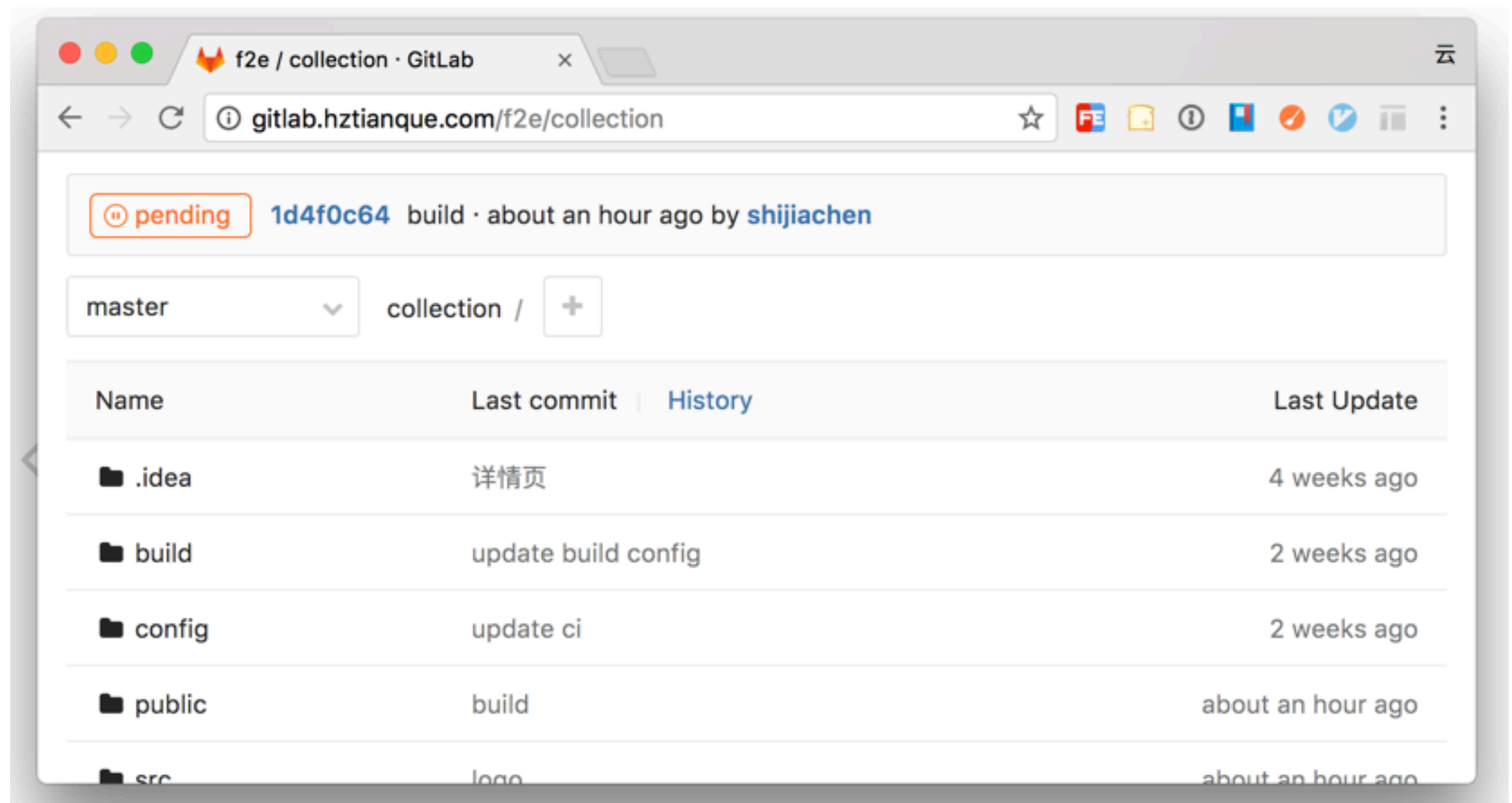
# sourcetree



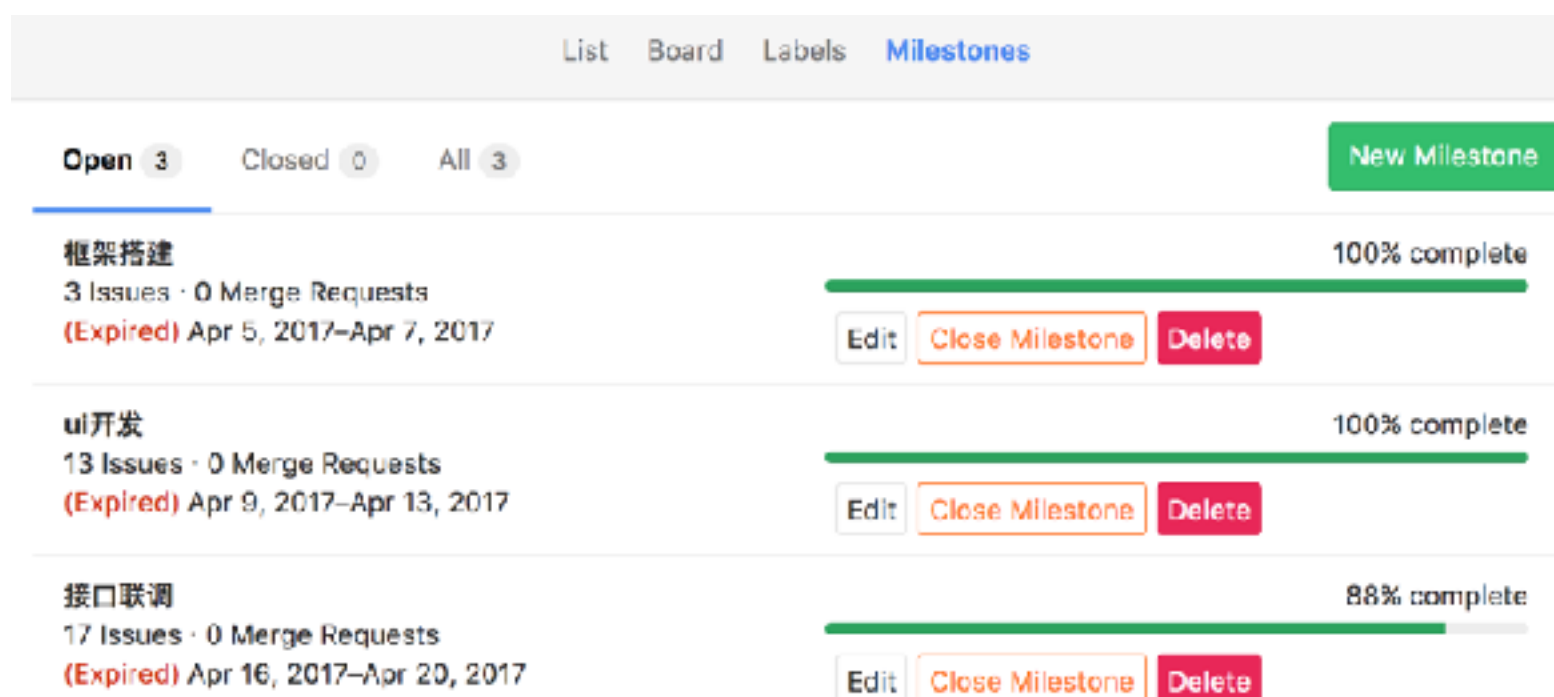
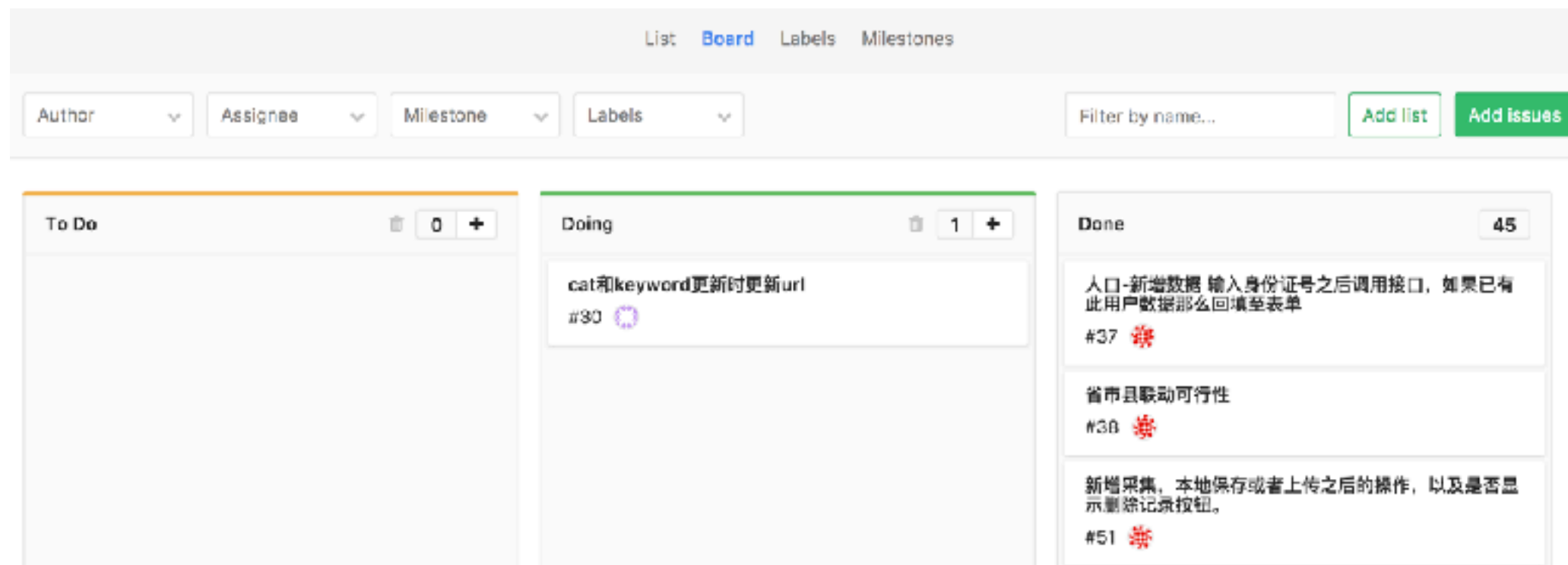
# gitflow



# gitlab



# 里程碑 & 看板



# 项目 & 组权限管理

- 访问权限
  - Private - 私有, 只有属于该项目成员才能clone
  - Internal - 内部, 有Gitlab账号的人都可以clone
  - Public - 公开, 任何人可以clone
- 行为权限
  - Guest - 访客
  - Reporter - 报告者; 可以理解为测试员、产品经理等
  - Developer - 开发者
  - Master - 主人, 负责对Master分支进行维护
  - Owner - 拥有者

# 发起pull request

# 创建一个开发分支

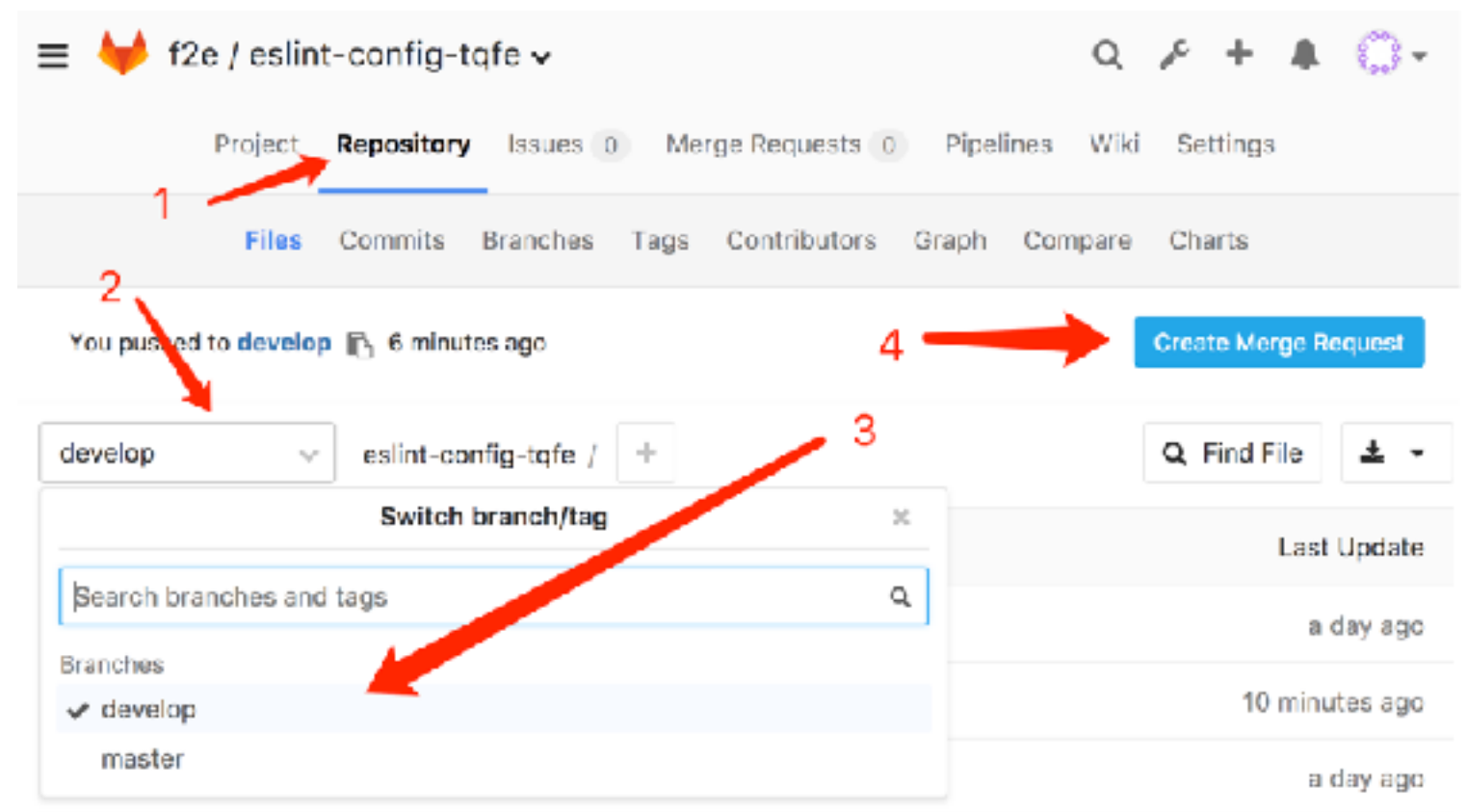
```
git checkout -b develop
```

# 开发新feature...

# 提交改动到本地和远端仓库

```
git add .
```

```
git commit -am 'some new feature'
```



# 处理pull request

## update rules

eslint webpack plugin在某些情况下识别空格是有bug

暂时去掉了代码缩进两个空格的校验

Edited less than a minute ago

Request to merge **develop** into **master**

Check out branch

Download as ▾

Accept Merge Request

☐ Remove source branch

☒ Modify commit message

You can also accept this merge request manually using the [command line](#).



0



0



Add

Discussion 0

Commits 1

**Changes 1**

Showing 1 changed file with 2 additions and 0

Hide whitespace changes

Inline

Side-by-side

deletions

▼ index.js



Edit

View file @f3286e4

```
...      @@ -17,6 +17,7 @@ module.exports = {
17      'array-bracket-spacing': ['error', 'never'],
18      'no-mixed-spaces-and-tabs': 2,
19      'no-multiple-empty-lines': 2,
20      + /* eslint 识别空格有bug, 暂时注释掉
```



# .gitignore

- 全局忽略
  - 用户家目录 ~/.gitignore
- 文件夹范围忽略
  - 文件夹下 ./gitignore

<https://github.com/github/gitignore>

23 lines (17 sloc) | 272 Bytes

```
1  # Compiled class file
2  *.class
3
4  # Log file
5  *.log
6
7  # BlueJ files
8  *.ctxt
9
10 # Mobile Tools for Java (J2ME)
11 .mtj.tmp/
12
13 # Package Files #
14 *.jar
15 *.war
16 *.ear
17 *.zip
18 *.tar.gz
19 *.rar
20
21 # virtual machine crash logs, see http://www.java.com
22 hs_err_pid*
```

# 地址问题

- git@gitlab.hztianque.com:f2e/docs.git
- ssh://git@gitlab.hztianque.com:22016/f2e/docs.git

# 参考资料

- [gitlab markdown语法](#)
- [coding markdown语法](#)
- [gitlab使用指南](#)
- [git简明指南](#)
- [添加key到gitlab](#)
- [创建pull request](#)

Q&A

THANK YOU