# Git

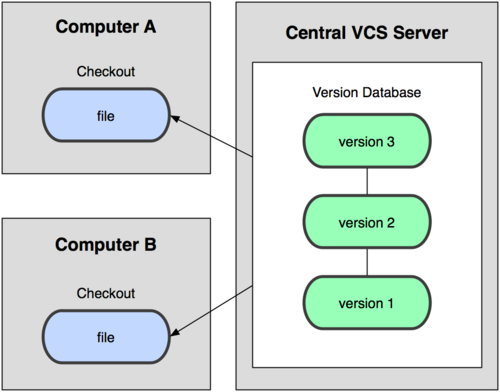
# 什么是Git

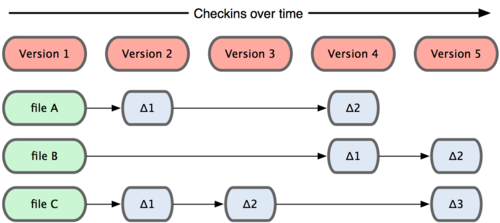
Git是一个版本控制工具

# Git和Svn区别在哪里?

## 从底层原理的角度

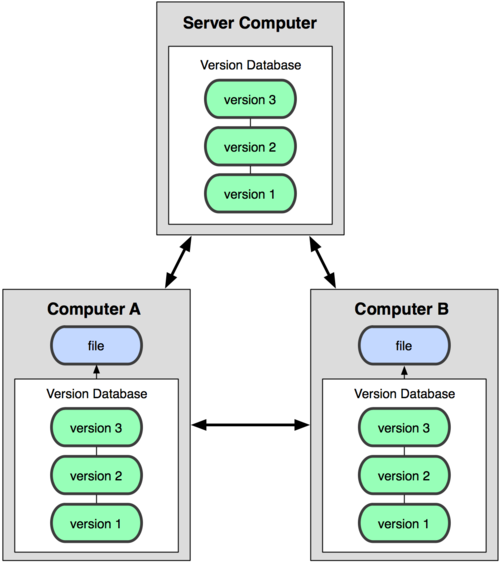
**SVN （CVCS）: SVN是集中化的版本控制系统，它有一个单一的集中管理的服务器，用于保存所有文件的修订版本。协同工作的人通过客户端.上传、更新文件或下载最新文件。**

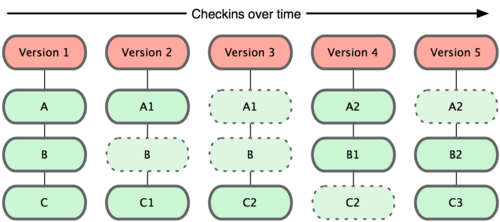




**Git：Git和Svn不同，Git是分布式版本控制系统，客户端并不只是拉取最新快照，而是将原始代码仓库镜像下来，这么一来，每个客户端都有一套完整的代码，如果git服务器发生故障，那么可以用客户端代码恢复。**

**Git 更像是把变化的文件作快照后，记录在一个微型的文件系统中。每次提交更新时，它会纵览一遍所有文件的指纹信息并对文件作一快照，然后保存一个指向这次快照 的索引。为提高性能，若文件没有变化，Git 不会再次保存，而只对上次保存的快照作一链接。**





## 从使用者的角度

1. **因为Git在客户端本地保存有一套完整的仓库，所以在无网络的情况下，使用者可以在本地提交代码文件。而SVN则只能在有网络的情况下提交代码。**
2. **git对程序源代码进行差异化的版本管理，代码库占极少的空间。易于代码的分支化管理。不支持中文，图形界面支持差，使用难度大，所以Git基本上只是开发人员使用。而SVN对中文支持好，操作简单，使用没有难度，美工人员，产品人员，测试人员，实施人员都可轻松上手。使用界面统一，功能完善，操作方便。**
3. **Git没有一个全局的版本号，而SVN有。**
4. **Git的内容完整性要优于SVN ：GIT的内容存储使用的是SHA-1哈希算法。这能确保代码内容的完整性，确保在遇到磁盘故障和网络问题时降低对版本库的破坏。**

# Git安装

## Github

### 安装客户端

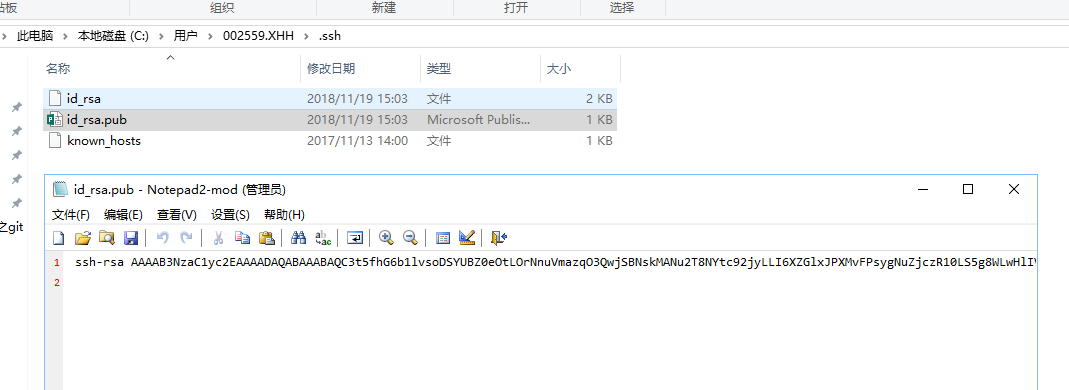
<https://gitforwindows.org/>

**首先安装好Github客户端。**

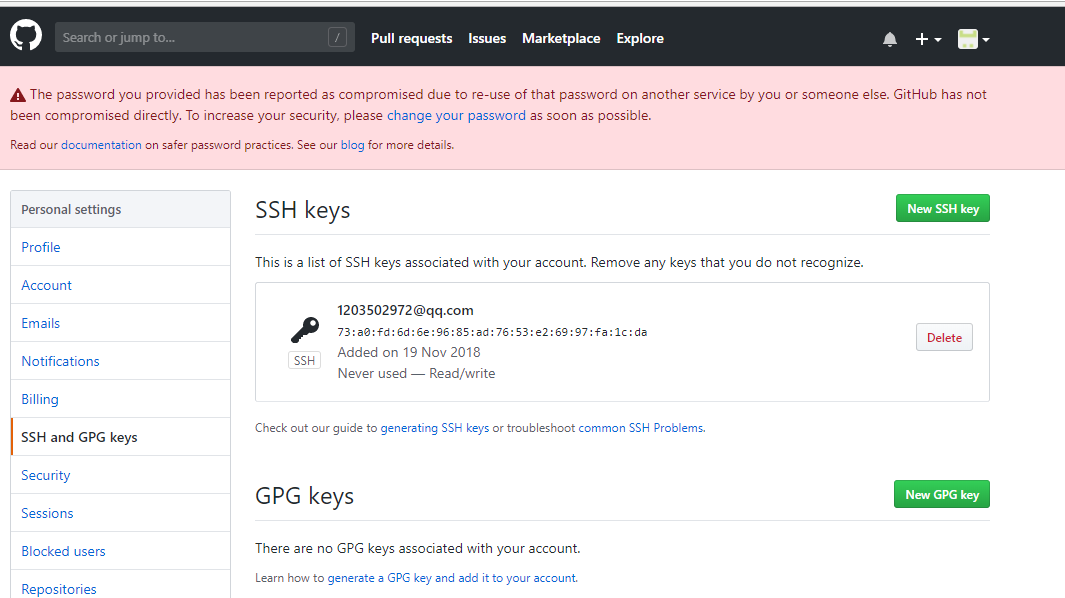
### 配置Git

#### 首先在本地创建ssh key

$ ssh-keygen -t rsa -C "1203502972@qq.com"

后面的1203502972@qq.com改为我们在github上注册的邮箱，之后会要求确认路径和输入密码，我们这使用默认的一路回车就行。成功的话会在~/下生成.ssh文件夹，进去，打开id\_rsa.pub，复制里面的key。

回到github上，进入 Account Settings（账户配置），左边选择SSH Keys，Add SSH Key,title随便填，粘贴在电脑上生成的key。



#### 测试连接

验证是否成功，在git bash下输入：

$ ssh -T git@github.com

如果是第一次的会提示是否continue，输入yes就会看到：You've successfully authenticated, but GitHub does not provide shell access 。这就表示已成功连上github。

#### 设置username和email

我们需要设置username和email，因为github每次commit都会记录他们。

$ git config --global user.name "your name"

$ git config --global user.email "your\_email@youremail.com"

#### 添加远程仓库地址

$ git remote add origin git@github.com:yourName/yourRepo.git

后面的yourName和yourRepo表示你再github的用户名和刚才新建的仓库，加完之后进入.git，打开config，这里会多出一个remote "origin"内容，这就是刚才添加的远程地址，也可以直接修改config来配置远程地址。

#### 检出仓库

git clone username@host:/path/to/repository

### 常用命令

#### 推送改动

如果已经将改动提交到了本地仓库，可以将改动推送到远程仓库，可以把 master 换成想要推送的任何分支。

git push origin master

如果还没有克隆现有仓库，并欲将仓库连接到某个远程服务器，可以使用如下命令添加：

git remote add origin <server>

如此就能够将改动推送到所添加的服务器上去了。

#### 分支

创建并切换到新分支：

git checkout -b new\_branch

切换分支：

git checkout branch

删除新分支：

git checkout -d branch

撤销操作

git checkout .

#### 更新与合并

获取并合并(fetch and merge):

git pull

其他分支与当前分支合并：

git merge <branch>

在这两种情况下，git 都会尝试去自动合并改动。遗憾的是，这可能并非每次都成功，并可能出现冲突（conflicts）。 这时候就需要你修改这些文件来手动合并这些冲突（conflicts）。改完之后，你需要执行如下命令以将它们标记为合并成功：

git add <filename>

在合并改动之前，可以使用如下命令预览差异：

git diff <source\_branch> <target\_branch>

#### 查看日志

git log

  --oneline 选项来查看历史记录的简洁的版本。

--graph 选项，查看历史中什么时候出现了分支、合并。

如果只想查找指定用户的提交日志可以使用命令：git log –author=name,

如果你要指定日期，可以执行几个选项：--since 和 --before，但是你也可以用 --until 和 --after。

$ git log --oneline --before={3.weeks.ago} --after={2010-04-18} --no-merges

#### 5. 回滚代码

服务器回滚：

git reset –hard commit-id<提价的id>

git push -f origin branch<分支>

如果只是本地做了修改，想放弃修改 ：

git checkout <filename>

1. 当执行 "git reset HEAD" 命令时，暂存区的目录树会被重写，被 master 分支指向的目录树所替换，但是工作区不受影响。
2. 当执行 "git rm --cached <file>" 命令时，会直接从暂存区删除文件，工作区则不做出改变。
3. 当执行 "git checkout ." 或者 "git checkout -- <file>" 命令时，会用暂存区全部或指定的文件替换工作区的文件。这个操作很危险，会清除工作区中未添加到暂存区的改动。
4. 当执行 "git checkout HEAD ." 或者 "git checkout HEAD <file>" 命令时，会用 HEAD 指向的 master 分支中的全部或者部分文件替换暂存区和以及工作区中的文件。这个命令也是极具危险性的，因为不但会清除工作区中未提交的改动，也会清除暂存区中未提交的改动。

#### 6. 标签

带注解的标签：

$ git tag -a v1.0

指定标签信息命令：

git tag -a <tagname> -m "runoob.com标签

查看标签：

$ git log --oneline --decorate --graph

如果我们忘了给某个提交打标签，又将它发布了，我们可以给它追加标签:

git tag -a v0.9 85fc7e7

# 4. Git服务器搭建

## 1. 下载安装包

添加gitlab镜像：

wget  <https://mirrors.tuna.tsinghua.edu.cn/gitlab-ce/yum/el7/gitlab-ce-10.0.0-ce.0.el7.x86_64.rpm>

## 安装

安装命令：

rpm -i gitlab-ce-10.0.0-ce.0.el7.x86\_64.rpm

## 修改gitlab配置文件指定服务器ip和自定义端口

vim  /etc/gitlab/gitlab.rb

例如修改端口号：



## 重置并启动GitLab

执行：

gitlab-ctlreconfigure

gitlab-ctl restart

## 常用命令

* sudo gitlab-ctl start # 启动所有 gitlab 组件；
* sudo gitlab-ctl stop # 停止所有 gitlab 组件；
* sudo gitlab-ctl restart # 重启所有 gitlab 组件；
* sudo gitlab-ctl status # 查看服务状态；
* sudo gitlab-ctl reconfigure # 启动服务；
* sudo vim /etc/gitlab/gitlab.rb # 修改默认的配置文件；
* gitlab-rake gitlab:check SANITIZE=true --trace # 检查gitlab；
* sudo gitlab-ctl tail # 查看日志；

# 5. 常见问题

**1. 对于push 100M文件限制的问题。**