git教程

Git的用法 Git Github 版本控制

简介

一个开源的分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目。与常用的版本控制工具 CVS, Subversion 等不同，它采用了分布式版本库的方式，不必服务器端软件支持。

Git与SVN区别

- **Git是分布的，SVN不是**
- **Git把内容按元数据方式存储，而SVN是按文件：把所有的资源控制系统都是把文件的元信息隐藏在一个类似.svn的文件夹里。**
- **Git分支与SVN的分支不同。**
- **Git没有一个全局的版本号，而SVN有。**
- **Git的内容完整性优于SVN：Git的内容存储使用的是SHA-1哈希算法，这能确保代码的完整性，确保在磁盘发生故障和网络问题是降低对版本库的破坏。**

Git快速入门

创建新仓库：git init
检出仓库：

Markdown 语法

- **创建一个本地仓库的克隆版本**：git clone /path/to/repository
- **克隆远程服务器上的版本**：git clone username@host：/path/torepository

添加和提交：

- **添加某文件到暂存区**：git add filename
- **添加工作目录中的所有文件到暂存区**：git add *

提交改动：git commit -m “代码提交信息”（将改动提交到了HEAD）

推送改动：

- **将改动提交到远程仓库**：git push origin master
- **如果没有克隆现有的仓库，想把自己的仓库链接到某个远程服务器**：git remote add origin xxXxxserver

分支

- **创建一个分支并切换过去**：git chechout -b branchName
- **切换回主分支**：git checkout master
- **删除某一分支**：git branch -d XxxbranchName
- **将分支推送到远程仓库，为他人所见**：git push origin branch

更新与合并

- **更新本地仓库到最新改动**：git pull
- **合并其他分支到当前分支（master）：git merge branch**
- **在合并之前先预览差异**：git diff source_branch target_branch

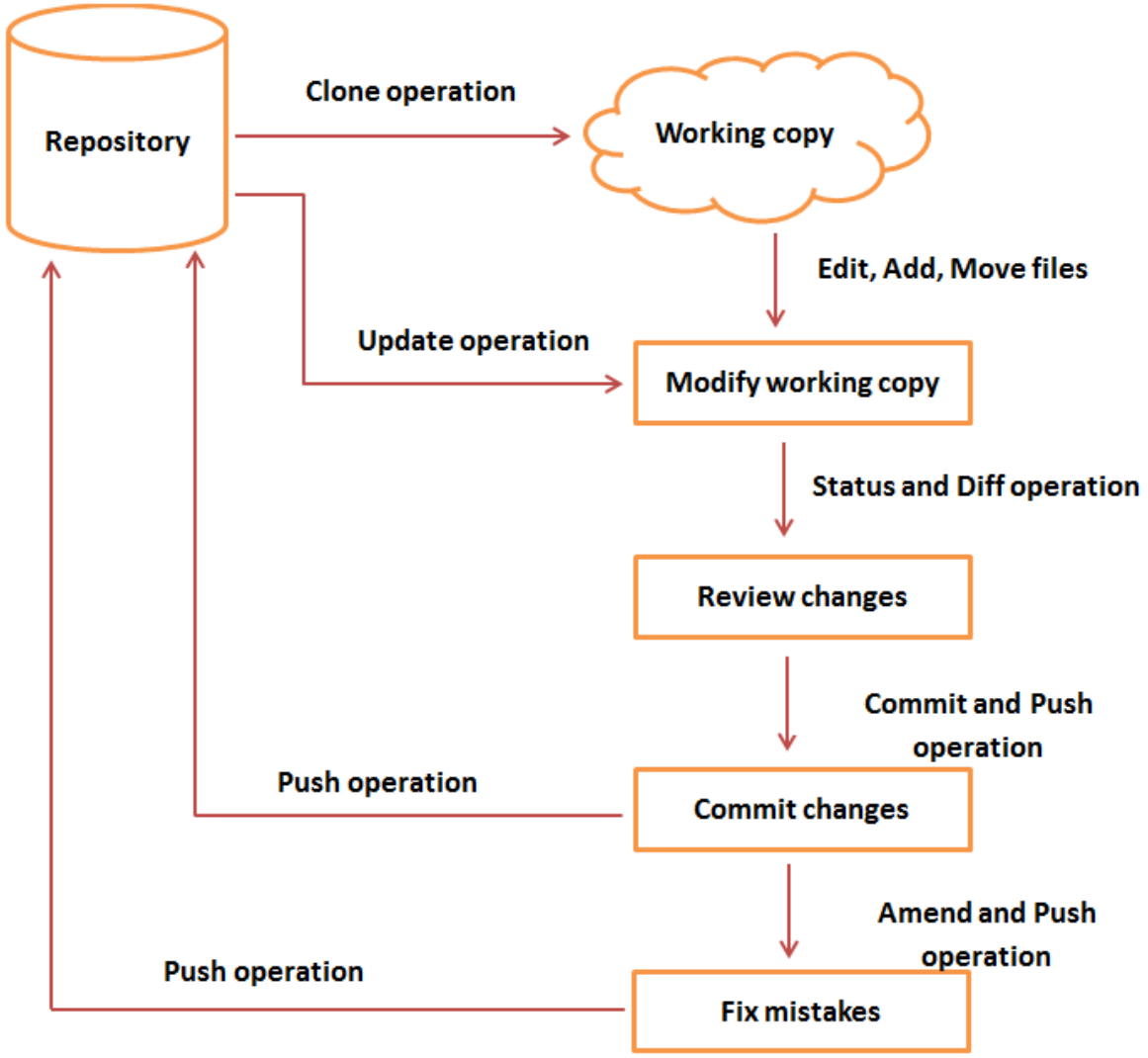
标签

- **提交之前先获取提交ID**：git log
- **创建一个叫做1.0.0的标签**：git tag 1.0.0 刚才通过log获取的ID前10位字符。

替换本地改动

- **如操作失误，可替换掉本地改动**：git checkout –filename
- **丢弃你在本地的所有改动与提交，可以到服务器上获取最新的版本历史，并将你本地主分支指向它**：git fetch origin git reset –hard origin/master

Git的工作流程



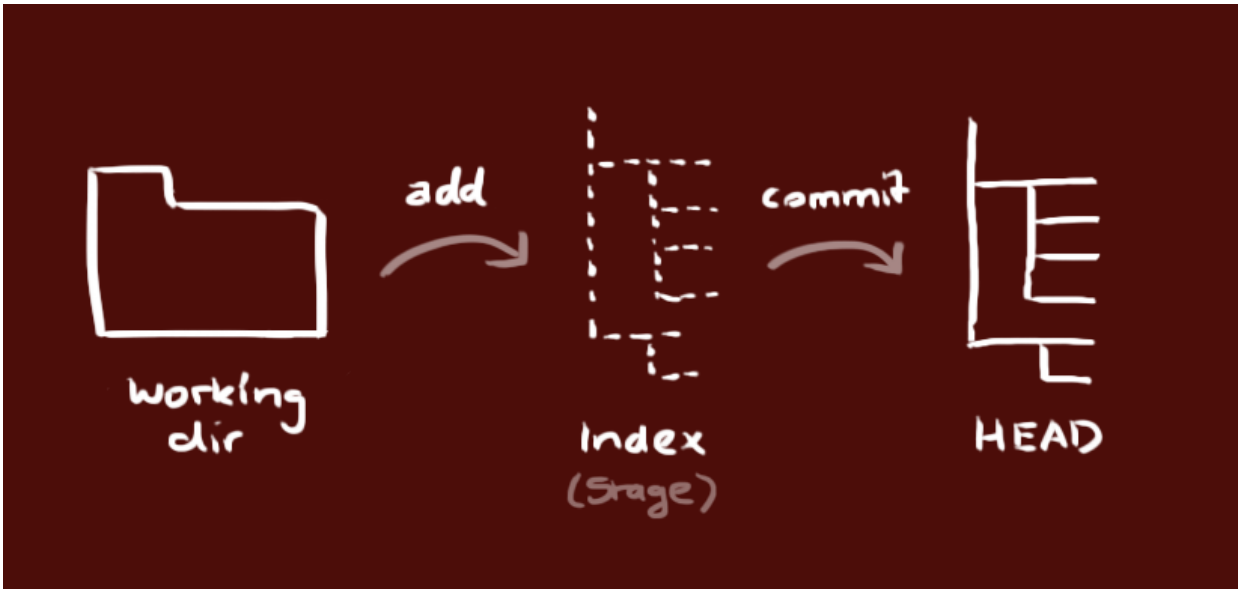
一般工作流程如下：

- **克隆Git资源为工作目录**
- **在克隆的资源上添加或修改文件**
- **如果其他人修改了，你可以更新资源**

- **在提交前查看修改**
- **提交修改**
- **在修改完成后，如果发现错误，可以撤回提交并再次修改并提交**

工作流

你的本地仓库由git维护的三颗"树"组成。第一个是你的`工作目录`，他持有实际文件；第二个是`暂存区`（index）,它像是缓存区域，临时保存你的改动；最后是`HEAD`,他指向你最后一次提交结果。



Git基本操作

获取与创建项目命令

- **git init**
  用 git init 在目录中创建新的 Git 仓库。 你可以在任何时候、任何目录中这么做，完全是本地化的。在目录中执行 git init，就可以创建一个 Git 仓库了。

```
$ mkdir QdhGithub
$ cd w3cschoolcc
$ git init
Initialized empty Git repository in /qdh/QdhGithub/.git/
# 在 /qdh/QdhGithub/.git/ 目录初始化空 Git 仓库完毕。
```

现在你可以看到在你的项目目录中有个 .git 的子目录。 这就是你的 Git 仓库了，所有有关你的此项目的快照数据都存放在这里。

```
ls -a
.   . .  ..      .git
```

-git clone
使用 git clone 拷贝一个 Git 仓库到本地，让自己能够查看该项目，或者进行修改。如果你需要与他人合作一个项目，或者想要复制一个项目，看看代码，你就可以克隆那个项目。 执行命令：git clone [url]

```
$ git clone git://github.com/schacon/simplegit.git
Initialized empty Git repository in /private/tmp/simplegit/.git/
remote: Counting objects: 100, done.
remote: Compressing objects: 100% (86/86), done.
remote: Total 100 (delta 35), reused 0 (delta 0)
Receiving objects: 100% (100/100), 9.51 KiB, done.
Resolving deltas: 100% (35/35), done.
$ cd simplegit/
$ ls
README   Rakefile lib
$ ls -a
.        ..       .git     README   Rakefile lib
$ cd .git
$ ls
HEAD        description info        packed-refs
branches    hooks       logs        refs
config      index       objects
```

基本快照
Git 的工作就是创建和保存你的项目的快照及与之后的快照进行对比。

- **git add**
  git add 命令可将该文件添加到缓存

```
$ git status -s
?? README
?? hello.java
$
```

git status 命令用于查看项目的当前状态。文件前面的问号表示还未将文件提交到缓存区
执行 git add 命令来添加文件：

```
$ git add README hello.java
```

再执行 git status命令查看文件状态

```
$ git status -s
A  README
A  hello.java
$
```

文件名前面的A表示已将其添加到缓存区

- **git diff**
  git diff 命令显示已写入缓存与已修改但尚未写入缓存的改动的区别。git diff 有两个主要的应用场景。
  - **尚未缓存的改动**：git diff
  - **查看已缓存的改动**：git diff –cached
  - **查看已缓存与未缓存的所有改动**：git diff HEAD
  - **显示摘要而非整个diff**：git diff –stat

```
$ git status -s
A  README
AM hello.java
$ git diff
diff --git a/hello.java b/hello.java
index e69de29..d1a9166 100644
--- a/hello.java
+++ b/hello.java
@@ -0,0 +1,3 @@
+<
+private String name;
+>
```

- **git commit**
  使用 git add 命令将想要快照的内容写入了缓存， 而执行 git commit 记录缓存区的快照。Git 为你的每一个提交都记录你的名字与电子邮箱地址，所以第一步需要配置用户名和邮箱地址。

```
$ git config --global user.name 'lyqdhgo'
$ git config --global user.email lyqdhgo@163.com
```

接下来我们写入缓存，并提交对 hello.java 的所有改动。在首个例子中，我们使用 -m 选项以在命令行中提供提交注释。

```
$ git add hello.java
$ git status -s
A  README
A  hello.java
$ git commit -m 'test comment from lyqdhgo'
[master (root-commit) 85fc7e7] test comment from lyqdhgo
 2 files changed, 4 insertions(+)
 create mode 100644 README
 create mode 100644 hello.java
```

现在我们已经记录了快照。如果我们再执行 git status:

```
$ git status
# On branch master
nothing to commit (working directory clean)
```

以上输出说明我们在最近一次提交之后，没有做任何改动，是一个”干净的工作目录”。

- **git reset HEAD**
  git reset HEAD 命令用于取消缓存已缓存的内容。

```
$ git status -s
 M README
 M hello.java
$ git add .
$ git status -s
M  README
M  hello.java
$ git reset HEAD -- hello.java
Unstaged changes after reset:
M hello.java
$ git status -s
M  README
```

- **git rm**
  git rm将文件从缓存区中移除

```
$ git rm hello.php
rm 'hello.php'
$ ls
README
```

默认情况下，git rm file 会将文件从缓存区和你的硬盘中（工作目录）删除。 如果要在工作目录中留着该文件，可以使用命令：

```
git rm --cached
```

Git的分支管理

创建分支：

```
git branch (branchname)
```

切换分支命令：

```
git checkout (branchname)
```

合并分支命令：

```
git merge
```

列出分支：

```
git branch
```

创建新分支并切换到该分支：

```
$ git checkout -b newtest
```

删除分支：

```
git branch -d (branchname)
```

Git查看提交历史

Git 提交了若干更新之后，想回顾下提交历史，我们可以使用 git log 命令查看
列出历史提交记录：

```
$ git log
commit 88afe0e02adcdfea6844bb627de97da21eb10af1
Merge: 14b4dca d7e7346
Author: lyqdhgo <lyqdhgo@163.com>
Date:   Sun Mar 1 15:03:42 2015 +0800
    Merge branch 'change_site'
    Conflicts:
        test.txt
```

还可以用 –oneline 选项来查看历史记录的简洁的版本

```
$ git log --oneline
88afe0e Merge branch 'change_site'
14b4dca 新增加一行
d7e7346 changed the site
556f0a0 removed test2.txt
2e082b7 add test2.txt
048598f add test.txt
```

还可以用 –graph 选项，查看历史中什么时候出现了分支、合并。以下为相同的命令，开启了拓扑图选项：

```
$ git log --oneline --graph
*   88afe0e Merge branch 'change_site'
|\
| * d7e7346 changed the site
* | 14b4dca 新增加一行
|/
* 556f0a0 removed test2.txt
* 2e082b7 add test2.txt
* 048598f add test.txt
```

Git标签

如果你达到一个重要的阶段，并希望永远记住那个特别的提交快照，你可以使用 git tag 给它打上标签。

```
# -a 选项意为"创建一个带注解的标签"。 它会记录这标签是啥时候打的，谁打的
#
$ git tag -a v1.0
```

查看标签：git log –decorate

```
$ git log --oneline --decorate --graph
*   88afe0e (HEAD, tag: v1.0, master) Merge branch 'change_site'
|\
| * d7e7346 (change_site) changed the site
* | 14b4dca 新增加一行
|/
* 556f0a0 removed test2.txt
* 2e082b7 add test2.txt
* 048598f add test.txt
* 85fc7e7 test comment from w3cschool.cc
```

如果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
$ git log --oneline --decorate --graph
*   88afe0e (HEAD, tag: v1.0, master) Merge branch 'change_site'
|\
| * d7e7346 (change_site) changed the site
* | 14b4dca 新增加一行
|/
* 556f0a0 removed test2.txt
* 2e082b7 add test2.txt
* 048598f add test.txt
* 85fc7e7 (tag: v0.9) test comment from lyqdhgo@163.com
```

查看所有标签

```
$ git tag
v0.9
v1.0
```

Git远程仓库

添加远程库

要添加一个新的远程仓库，可以指定一个简单的名字，以便将来引用,命令格式如下：

```
git remote add [shortname] [url]
```

由于你的本地Git仓库和GitHub仓库之间的传输是通过SSH加密的，所以我们需要配置验证信息：
使用以下命令生成SSH Key：

```
#邮箱为github注册邮箱
$ ssh-keygen -t rsa -C "qdhwdnda21k@yahoo.com"
```

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，粘贴在你电脑上生成的key。
为了验证是否成功，输入以下命令：

```
$ ssh -T git@github.com
#出现该信息说明我们已成功连上 Github。
Hi tianqixin! You've successfully authenticated, but GitHub does not
provide shell access.
```

登录github后点击” New repository ”
之后在在Repository name 填入 TestRepository (远程仓库名)，其他保持默认设置，点击”Create repository”按钮，就成功地创建了一个新的Git仓库：
把本地仓库的内容推送到GitHub仓库

```
$ git remote add origin git@github.com:Morcal /TestRepository.git
$ git push -u origin master
Counting objects: 21, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (15/15), done.
Writing objects: 100% (21/21), 1.73 KiB | 0 bytes/s, done.
Total 21 (delta 4), reused 0 (delta 0)
To git@github.com:Morcal /TestRepository.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

查看当前远程库

```
$ git remote
origin
$ git remote -v
origin  git@github.com:Morcal /TestRepository.git
```

提取远程仓库

- **从远程仓库下载新分支**：git fetch
  执行完后需要执行git merge远程分支到你所在的分支
- **从远程仓库提取数据并合并到当前分支**：git pull

推送到远程仓库：git push [alias] [branch]
以上命令就是将你的[branch]分支推送到[alias]远程仓库的[branch]分支
删除远程仓库：git remote rm [别名]

```
$ git remote -v
origin  git@github.com:tianqixin/w3cschool.cc.git (fetch)
origin  git@github.com:tianqixin/w3cschool.cc.git (push)
$ git remote add origin2 git@github.com:tianqixin/w3cschool.cc.git
$ git remote -v
origin  git@github.com:tianqixin/w3cschool.cc.git (fetch)
origin  git@github.com:tianqixin/w3cschool.cc.git (push)
origin2 git@github.com:tianqixin/w3cschool.cc.git (fetch)
origin2 git@github.com:tianqixin/w3cschool.cc.git (push)
$ git remote rm origin2
$ git remote -v
origin  git@github.com:tianqixin/w3cschool.cc.git (fetch)
origin  git@github.com:tianqixin/w3cschool.cc.git (push)
```

总结

将本地文件提交到Github上

```
 #创建本地ssh
ssh-keygen -t rsa -C "qdhwdnda21k@yahoo.com"
 #用记事本打开 id_rsa.pub 文件，将该文件中的全部内容复制添加到github的SSH Keys
中
 #验证是否配置成功
ssh -T git@github.com
 #如果出现Hi Morcal! You've successfully authenticated, but GitHub does
not provide shell access. 就说明配置成功，可以连接上GitHub;
 #配置本地用户和邮箱
git config --global user.name "lyqdhgo" //设置用户名
git config --global user.email "lyqdhgo@163.com"  //设置邮箱
git init
git status -s
git add README.md
git status
 #出现nothing to commit (working directory clean) 说明git add完成，已没有
向缓存区提交的了
git commit -m "first commit"
git remote add origin git@github.com:han1202012/TabHost_Test.git
 #此处容易出现fatal: remote origin already exists的错误，解决方案如下
git remote rm origin //先移除掉
git remote add origin git@github.com:han1202012/TabHost_Test.git //再添
加
git push -u origin master
```