参考网站：

<https://www.liaoxuefeng.com/>

Git简介

Git是目前世界上最先进的**分布式 版本控制系统**（没有之一）。

集中式版本控制系统，版本库是集中存放在中央服务器；最大的毛病就是必须联网才能工作；

Git安全性要高很多，因为每个人电脑里都有完整的版本库；强大的分支管理

可视化工具

1.Pycharm里面的git插件

2.自带的git GUI

3....

常见概念

工作区、暂存区、分支；

版本、管理修改（撤销修改、删除修改）、删除文件；

远程仓库（添加远程仓库、从远程仓库克隆）；

分支管理（创建与合并、解决冲突、Bug分支、Feature分支、多人协作、Rebase)；

标签管理（创建标签、操作标签）

自定义Git（忽略特殊文件、配置别名、搭建Git服务器）

分支概念

每次提交，Git都把它们串成一条时间线，这条时间线就是一个分支。

Git用master指向最新的提交，再用HEAD指向master，就能确定当前分支，以及当前分支的提交点。



# Git 核心概念

Git 最核心的一个概念就是工作流。

工作区(Workspace)是电脑中实际的目录。

暂存区(Index)类似于缓存区域，临时保存你的改动。

仓库区(Repository)，分为本地仓库和远程仓库。

# git代码提交的步骤：

git add从工作区提交到暂存区

git commit从暂存区提交到本地仓库

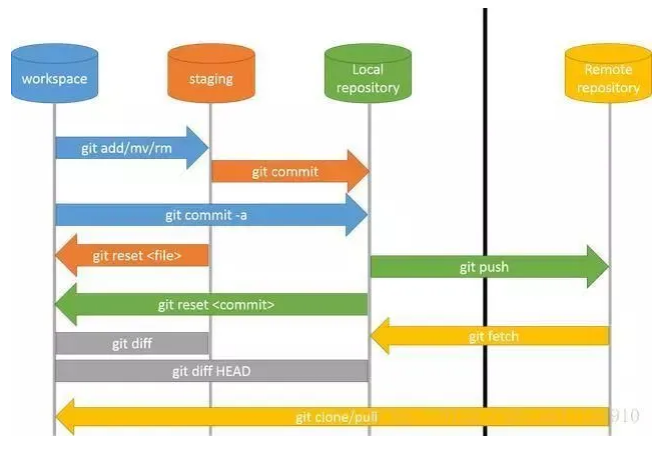
git push或git svn dcommit从本地仓库提交到远程仓库

git push的一般形式为 git push <远程主机名> <本地分支名>  <远程分支名> ，例如 git push origin master：refs/for/master ，即是将本地的master分支推送到远程主机origin上的对应master分支， origin 是远程主机名

eg:

git push -u origin master

一般来说，记住以下命令，便可进行日常工作了（图片来源于网络）：



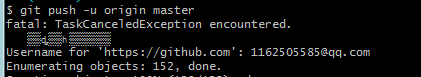
# git commit错误

error: pathspec 'project'' did not match any file(s) known to git.

将单引号修改为双引号

添加-m参数

# git push错误



解决方法: 输入GitHub的邮箱, 注意此处只能输入邮箱, 输入GitHub上的userName会出错, 此处是一个巨大的坑.

Already up to date. 错误

一般在进行git操作的时候出现Already up to date.问题有两种情况

　　一、当前分支的文件和远程仓库的文件不统一，这时候就需要将远程仓库的文件同步到本地仓库后再merge

　　　　git pull origin 分支名   　　　　　　#  更新本地仓库数据

　　　　git merge　要合并的分支名　　　　# 重新合并分支

　　　　git push origin 远程分支名　　　　　　#  提交合并后的数据

　　二、本地仓库有新增的文件没有add和commit

　　　　git add .

　　　　git commit -m "描述信息"

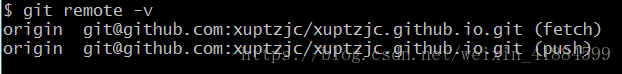
　　　　git merge　要合并的分支名　　　　# 重新合并分支

　　　　git push origin 远程分支名　　　　　　#  提交合并后的数据

# 本地仓库更新

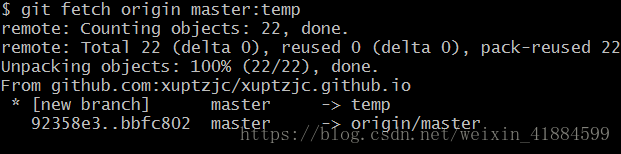
查看远程分支

使用 git remote -v 可以查看远程仓库分支



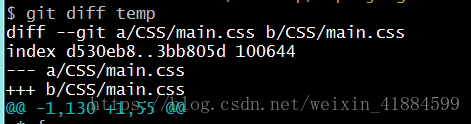
从远程获取最新版本到本地

使用git fetch origin master:temp 在本地新建一个temp分支，并将远程仓库的master分支的代码下载到本地分支上面



比较temp与本地仓库

使用 git diff temp 可以比较当前本地仓库和temp仓库之间的

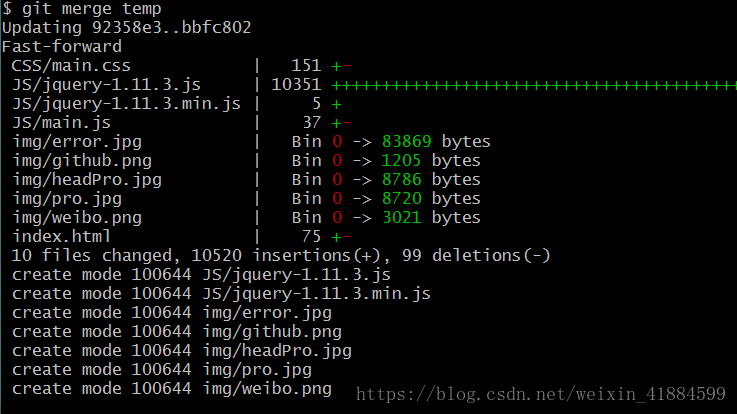


此过程时间比较长, 容易引起人的误会, 以为程序以及执行完毕, 卡在了显示差异的界面, 其实是差异比较还未结束, 请耐心等待, 直到出现可以重新输入命令的时候.

更新本地仓库

在确定修改了之后我们就可以更新本地仓库了

使用 git merge temp就可以更新当前的仓库了，更新之后我们把创建的temp文件夹删除就可以了.



# 删除本地仓库分支

#查看，发现有本地分支temp

cc@cc-PC MINGW64 /f/my-center/cloud/server-services-config (temp)

$ git branch -a

master

\* temp

remotes/origin/HEAD -> origin/master

remotes/origin/master

#需要先切换到目标之外的分支，才能删除目标分支

cc@cc-PC MINGW64 /f/my-center/cloud/server-services-config (temp)

$ git checkout master

Switched to branch 'master'

M weshow-sqs-consumer-task/pom.xml

Your branch is up to date with 'origin/master'.

#如还有未被提交的change，执行如下命令会有错误提示

cc@cc-PC MINGW64 /f/my-center/cloud/server-services-config (master)

$ git branch -d temp

error: The branch 'temp' is not fully merged.

If you are sure you want to delete it, run 'git branch -D temp'.

#需要用如下命令进行删除

cc@cc-PC MINGW64 /f/my-center/cloud/server-services-config (master)

$ git branch -D temp

Deleted branch temp (was 95447d5).

常用命令

创建SSH Key：

$ ssh-keygen -t rsa -C "youremail@example.com"

在用户主目录里找到.ssh目录，把id\_rsa.pub发给我，我再写到服务器上，这样你就能参与变成了。

vim 查看id\_rsa.pub

退出vim编辑器

1. 按ESC键
2. shift+:
3. 输入”q!”, 即可退出vim编辑器

初始化本地仓库(这是你的工作区）

git init

一个目录，这个目录里面的所有文件都可以被Git管理起来，每个文件的修改、删除，Git都能跟踪，以便任何时刻都可以追踪历史，或者在将来某个时刻可以“还原”。

不要使用Windows自带的记事本编辑任何文本文件，Notepad++可以代替。（默认编码设置为UTF-8 without BOM）

将本地仓库与远程仓库关联

git remote add origin git@10.87.26.159:/srv/sample.git

。。。。。。

没有git init的话，直接克隆

git clone git@10.87.26.159:/srv/sample.git

再建立本地分支。

将文件添加至**暂存区（概念）**

git add <file>

撤销修改

（把文件在**工作区（概念）**的修改全部撤销）

1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令

git checkout --<file>

2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，

第一步用命令git reset HEAD <file>，就回到了场景1，

第二步按场景1操作。

3：已经提交了不合适的修改到版本库时，想要撤销本次提交（当前版本回退上一个版本）

git reset --hard HEAD^

tips: HEAD 表示当前版本，上一个版本是HEAD^，上上个版本是HEAD^^

提交文件至本地仓库

（也就是提交到当前分支，可能是dev，也可能是你新建的youname分支）

git commit -m "描述的提交的信息"

查看仓库状态

git status

查看历史提交记录

git log

查看历史提交记录(精简的信息）

git log --pretty=oneline

显示输入命令的历史记录

git reflog

删除文件

rm filename.txt

从版本库中删除文件

git rm <filename>

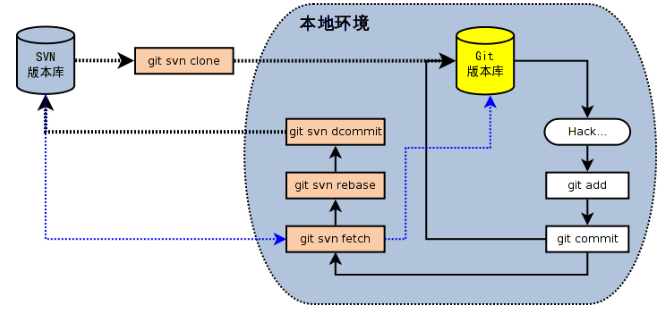
git commit -m "....."

=======================================

# Git-SVN整合

若服务器使用的 SVN，但是本地想要体验 Git 的本地分支，离线操作等功能，可以使用 Git-SVN功能。

常用操作如下（图片来源于网络）：



## ****[ Git-SVN ]功能****

# 下载一个 SVN 项目和它的整个代码历史，并初始化为 Git 代码库  
$ git svn clone -s [repository]  
# 查看当前版本库情况  
$ git svn info  
# 取回远程仓库所有分支的变化  
$ git svn fetch  
# 取回远程仓库当前分支的变化，并与本地分支变基合并  
$ git svn rebase  
# 上传当前分支的本地仓库到远程仓库  
$ git svn dcommit  
# 拉取新分支，并提交到远程仓库  
$ svn copy [remote\_branch] [new\_remote\_branch] -m [message]  
# 创建远程分支对应的本地分支  
$ git checkout -b [local\_branch] [remote\_branch]

## 四、初始化

从本节开始，除特殊说明，以下命令均适用于 Git 与 Git-SVN。  
  
# 在当前目录新建一个Git代码库  
$ git init  
# 下载一个项目和它的整个代码历史 [Git only]  
$ git clone [url]

## 五、配置

# 列举所有配置  
$ git config -l  
# 为命令配置别名  
$ git config --global alias.co checkout  
$ git config --global alias.ci commit  
$ git config --global alias.st status  
$ git config --global alias.br branch  
# 设置提交代码时的用户信息  
$ git config [--global] user.name "[name]"  
$ git config [--global] user.email "[email address]"

Git 用户的配置文件位于 ~/.gitconfig

Git 单个仓库的配置文件位于 ~/$PROJECT\_PATH/.git/config

## 六、增删文件

# 添加当前目录的所有文件到暂存区  
$ git add .  
# 添加指定文件到暂存区  
$ git add <file1> <file2> ...  
# 添加指定目录到暂存区，包括其子目录  
$ git add <dir>  
# 删除工作区文件，并且将这次删除放入暂存区  
$ git rm [file1] [file2] ...  
# 停止追踪指定文件，但该文件会保留在工作区  
$ git rm --cached [file]  
# 改名文件，并且将这个改名放入暂存区  
$ git mv [file-original] [file-renamed]

把文件名 file1 添加到 .gitignore 文件里，Git 会停止跟踪 file1 的状态。

## 七、分支

# 列出所有本地分支  
$ git branch  
# 列出所有本地分支和远程分支  
$ git branch -a  
# 新建一个分支，但依然停留在当前分支  
$ git branch [branch-name]  
# 新建一个分支，并切换到该分支  
$ git checkout -b [new\_branch] [remote-branch]  
# 切换到指定分支，并更新工作区  
$ git checkout [branch-name]  
# 合并指定分支到当前分支  
$ git merge [branch]  
# 选择一个 commit，合并进当前分支  
$ git cherry-pick [commit]  
# 删除本地分支，-D 参数强制删除分支  
$ git branch -d [branch-name]  
# 删除远程分支  
$ git push [remote] :[remote-branch]

## 八、提交

# 提交暂存区到仓库区  
$ git commit -m [message]  
# 提交工作区与暂存区的变化直接到仓库区  
$ git commit -a  
# 提交时显示所有 diff 信息  
$ git commit -v  
# 提交暂存区修改到仓库区，合并到上次修改，并修改上次的提交信息  
$ git commit --amend -m [message]  
# 上传本地指定分支到远程仓库  
$ git push [remote] [remote-branch]

## 九、拉取

# 下载远程仓库的所有变动 (Git only)  
$ git fetch [remote]  
# 显示所有远程仓库 (Git only)  
$ git remote -v  
# 显示某个远程仓库的信息 (Git only)  
$ git remote show [remote]  
# 增加一个新的远程仓库，并命名 (Git only)  
$ git remote add [remote-name] [url]  
# 取回远程仓库的变化，并与本地分支合并，(Git only), 若使用 Git-SVN，请查看第三节  
$ git pull [remote] [branch]  
# 取回远程仓库的变化，并与本地分支变基合并，(Git only), 若使用 Git-SVN，请查看第三节  
$ git pull --rebase [remote] [branch]

## 十、撤销

# 恢复暂存区的指定文件到工作区  
$ git checkout [file]  
# 恢复暂存区当前目录的所有文件到工作区  
$ git checkout .  
# 恢复工作区到指定 commit  
$ git checkout [commit]  
# 重置暂存区的指定文件，与上一次 commit 保持一致，但工作区不变  
$ git reset [file]  
# 重置暂存区与工作区，与上一次 commit 保持一致  
$ git reset --hard  
# 重置当前分支的指针为指定 commit，同时重置暂存区，但工作区不变  
$ git reset [commit]  
# 重置当前分支的HEAD为指定 commit，同时重置暂存区和工作区，与指定 commit 一致  
$ git reset --hard [commit]  
# 新建一个 commit，用于撤销指定 commit  
$ git revert [commit]  
# 将未提交的变化放在储藏区  
$ git stash  
# 将储藏区的内容恢复到当前工作区  
$ git stash pop

# **十一、查询**

# 查看工作区文件修改状态  
$ git status  
# 查看工作区文件修改具体内容  
$ git diff [file]  
# 查看暂存区文件修改内容  
$ git diff --cached [file]  
# 查看版本库修改记录  
$ git log  
# 查看某人提交记录  
$ git log --author=someone  
# 查看某个文件的历史具体修改内容  
$ git log -p [file]  
# 查看某次提交具体修改内容  
$ git show [commit]

## ****导航 —— 跳到之前的分支****

git checkout -

## ****查看历史****

*# 每个提交在一行内显示*  
git log *--oneline*  
  
*# 在所有提交日志中搜索包含「homepage」的提交*  
git log *--all --grep='homepage'*  
  
*# 获取某人的提交日志*  
git log *--author="Maxence"*

## [哎呀：之前重置了一个不想保留的提交，但是现在又想要回滚？](http://mp.weixin.qq.com/s?__biz=MzI2MTIzMzY3Mw==&mid=2247489903&idx=1&sn=ab042bcba27e90b5e636fbc5b12392a3&chksm=ea5cd409dd2b5d1ff96ab50afa73d9dce4b308b1ee99eb903e84c48cc600d727d0626b80eefb&scene=21" \l "wechat_redirect" \t "https://mp.weixin.qq.com/_blank)

*# 获取所有操作历史*  
git reflog  
  
*# 重置到相应提交*  
git **reset** HEAD@{4}  
*# ……或者……*  
git **reset** --hard <提交的哈希值>

## ****哎哟：我把本地仓库搞得一团糟，应该怎么清理？****

git fetch origin  
git checkout **master**  
**git** reset --hard origin/master

## ****查看我的分支和 master 的不同****

git diff master..**my**-branch

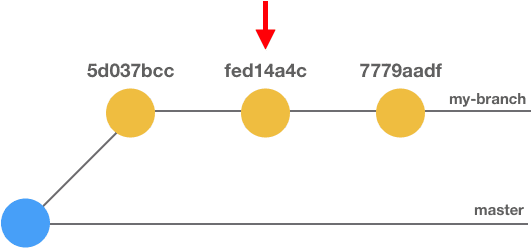
## ****定制提交****

**# 编辑上次提交**  
git commit --amend -m "更好的提交日志"  
  
**# 在上次提交中附加一些内容，保持提交日志不变git add . && git commit --amend --no-edit**  
  
**# 空提交 —— 可以用来重新触发 CI 构建**  
git commit --allow-empty -m "chore: re-trigger build"

**squash 提交**比方说我想要 rebase 最近 3 个提交：  
- git rebase -i HEAD~3  
- 保留第一行的 pick，剩余提交替换为 squash 或 s  
- 清理提交日志并保存（vi 编辑器中键入 :wq 即可保存）

pick 64d26a1 feat: add index.js  
s 45f0259 **fix**: **update** index.js  
s 8b15b0a **fix**: typo in index.js

## ****修正****

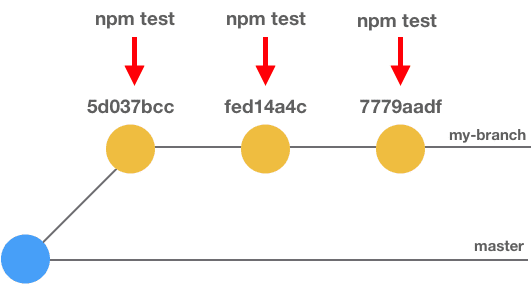
比方说想在提交 fed14a4c 加上一些内容。  
**git 提交分支**

git add .  
git **commit** *--fixup HEAD~1*  
# 或者也可以用提交的哈希值（fed14a4c）替换 **HEAD**~1  
  
git rebase -i **HEAD**~3 *--autosquash*  
# 保存并退出文件（VI 中输入 `:wq`）

## ****rebase 的时候在每个提交上执行命令****

如果特性很多，一个分支里可能有多个提交。如果测试失败了，你希望能找到导致测试失败的提交。这时候你可以使用 rebase --exec 命令在每个提交上执行命令。

# 在最近 3 个提交上运行 `npm test` 命令  
git rebase HEAD~3 --exec "npm test"



## ****暂存****

暂存不止是 git stash 和 git stash pop ;)

*# 保存所有正在追踪的文件*  
git stash save "日志信息"  
  
*# 列出所有的暂存项*  
git stash list  
  
*# 获取并删除暂存项*  
git stash apply stash@{1}  
git stash drop stash@{1}  
*# ……或使用一条命令……*  
git stash **pop** stash@{1}

## ****清理 远程仓库不存在的分支****

# 移除远程仓库上不存在的分支  
git fetch -p  
  
# 移除所有包含 `greenkeeper` 的分支  
git fetch -p && git branch --remote | fgrep greenkeeper | sed 's/^.\{9\}//' | xargs git push origin --delete

## ****GitHub = Git + Hub****

我把 Hub 当成 git 的一个封装来用。你如果也想这么做，可以设置一个别名：alias git='hub'

**# 打开浏览器访问仓库 url（仅限 GitHub 仓库）git browse**

[额外福利：我最喜爱的 git 别名](http://mp.weixin.qq.com/s?__biz=MzI2MTIzMzY3Mw==&mid=2247489890&idx=1&sn=bb9323d83eb7320840e5d23082ec8aee&chksm=ea5cd404dd2b5d12e64d385fb8003f9da99d34d0ecfa648147aca44b59034264d61d8c334667&scene=21" \l "wechat_redirect" \t "https://mp.weixin.qq.com/_blank)

alias g='git'  
alias glog='git log --oneline --decorate --graph'  
alias gst='git status'  
alias gp='git push'  
alias ga='git add'alias gc='git commit -v'  
  
*#*  
alias yolo='git push --force'  
  
*# 每周站会汇报工作时用*  
git-**standup**() {  
 AUTHOR=${AUTHOR:="`git config user.name`"}  
  
 since=yesterday  
 **if** [[ $(date +%u) == 1 ]] ; **then**  
 since="2 days ago"  
 **fi**  
  
 git log --all --since "$since" --oneline --author="$AUTHOR"  
}