

# Git 常用命令

## 1. git 是什么

Git 是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。

Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。

Git 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。

## 2. Git 与 SVN 的区别

Git 不仅仅是个版本控制系统，它也是个内容管理系统(CMS)，工作管理系统等。

如果你是一个具有使用 SVN 背景的人，你需要做一定的思想转换，来适应 Git 提供的一些概念和特征。

Git 与 SVN 区别点：

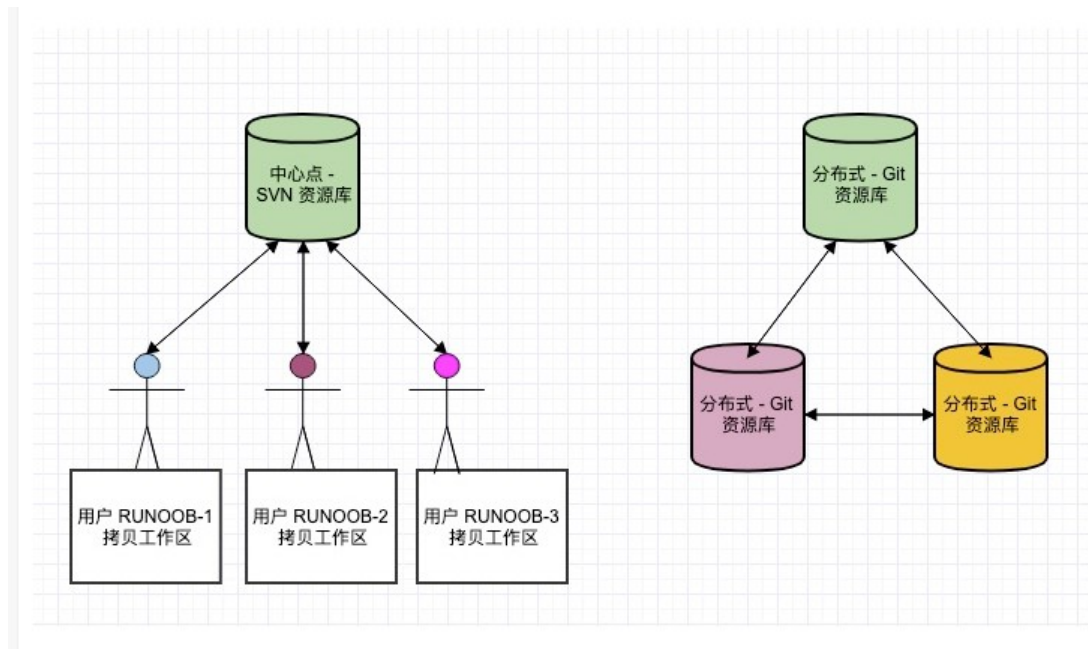
1、Git 是分布式的，SVN 不是：这是 Git 和其它非分布式的版本控制系统，例如 SVN，CVS 等，最核心的区别。

2、Git 把内容按元数据方式存储，而 SVN 是按文件：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、.cvs 等的文件夹里。

3、Git 分支和 SVN 的分支不同：分支在 SVN 中一点都不特别，其实它就是版本库中的另外一个目录。

4、Git 没有一个全局的版本号，而 SVN 有：目前为止这是跟 SVN 相比 Git 缺少的最大的一个特征。

5、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。



### 3. GIT 常用命令

4.

创建分支

`git branch test`

切换分支

`git checkout test`

创建并切换分支

`git checkout -b test`

删除远程分支

`git push origin --delete test`

更新删除的分支信息

`git fetch -p`

删除本地分支

`git branch -D test`

显示分支

`git branch -a`

添加到的暂存区

```
git add a.txt b.txt
```

提交

```
git commit -m "test test"
```

把 test 分支上传远程

```
git push origin test
```

把远程分支下载到本地并切换到 test 分支

```
git checkout -t origin/test
```

查看 git 版本

```
git --version
```

查看状态

```
git status
```

git 查看用户信息

```
git config user.name
```

```
git config user.email
```

```
git config --list
```

合并分支(切换到 master)

```
git merge test
```

然后再提交

```
git push
```

删除远程文件(或本地工作文件)

```
git rm test.txt
```

```
git commit
```

```
git push origin master
```

## 查看日记

git log --pretty=oneline 显示一条线

```
commit 2c1efb3f659a8cbdcde2610ca89242580ed07 (HEAD -> editor, origin/editor) 放大图片样式修改
645ff38931f33733d454201a43325cae932ec201 评论删除默认展示图片
4ed852693e0d947fef0f8093d1696e594ba24706 详情页图片放大功能
b7abceee189a5bb2ee2ef75027da64f65271a2e5 详情页图片展示
533ae78d05b053a5faee270ebecbc246e316af6b 发表文章修改
676b3b0275cab9e2b9309197b752271a16179cbb (origin/master, origin/HEAD, master) 线上发布版1
f31c9bff4f4fb23d79f6d5c8c497a9417bf95af3 修改保存草稿必填项（发布版）
1407360ec8ad99e0737c1c55aae24fe4f27d7bd7 添加发布文章页面
4c2a594b7c7f62801be556ce863775d4e2fb369b 编辑框未分开
b2193b221267edbc8ff88aa8ea63912426e75024 调试编辑框
527d9b21057a6a8624cf1a265913df1ac239d0d1 付费视频暂停bug
1fd8760571ceb3b868b61fdfe6b433604c416206 视频播放bug
e5ec41808a878d33182781bc00e037377092f7c5 bug修改-
96bf010524ac211d5cc3ca77f128682c4d8078fa 超链接修改
8838aba1929ed495592217494376fb030b74eb6f 修改展示
d5d14be0deaae3707f5f799ada56ea2f1a05d4b 回滚代码回滚问题
8a1ef7ca42cc69d86058a6a6300ed31126ab6687 详情预览
5ab21d96db0d6234d040451da340edb7724f02eb 详情展示修改
1894a79ede162082a309cbb4ff01a8fb8c78d1a7 bug修改
bd9dc8de1665b924cb195a6b4759ae5fa8687bee 详情修改
3f26230de723f49c315d65c95a64c34b27acae3 bug修改
3adc7c834f88399aa6baf87adda3f1628b8f2a09 调试代码
e10765fb1615e725570d170d9a682e36cb21ab4a 评论调试
e838689dc6d6f71c6f8f9d691b5f97560d7e0fca 详情页展示
919e1af74941653017831c4332bb57c118c87214 2端联调
834a2bcb94d42b4789909f03327e85fde5f16fa4 调试接口与h5对接
fd3c0ed2eb79d37023caa81eb8aff682185cd31e 调试接口代码
b552745efa8f7af9fb2994e36ef40ae2874de161 页面样式调整
6522cda9100f193853200c2466bf7a759d582940 更新插件
4a28ed47f4dc41b2685eb97cf392372b7213a7e65 发布文章样式修改
c0b7f0d28b0ca0a369adaaa987c94aef2181224 详情页展示
b6e771f6d90ed83046094390faf32566d9554e7e 详情页展示
c69dc635a0325b13d3cfa1490dc4db70a4e45294 详情展示页
45910980409032d27c0fe4240474f061423b8005 发布文章
3b3d046b80551ca8cd25b20cd27a949580f58aac zx-editor
1d7b4c54550d2df8f29b45d3b9ac3a58899e6b4a 详情页信息
b336a3bbe10708d26bb238b0b8b93148729b90c 文章详情展示页面
e664df1e1796bd0bd57578a8f4b7c2556bc90a2f 新增editor模块
b219b39bbe5f5db6f64dab62b8c97033795ee6dc 暂无文件提示去掉
700ab0e1b0875dc22ebd8946b0f90a84c6b11477 修改bug
2cbf8c8f3fcc90f0866db258a4abb96e5a424b 下载添加接口处理
8f9d48fa31a12485977747d0d1b34823ee8d0bc3 文件夹图标修改，取消全选改为取消
5448b7f491e43fc7fdbca12512076276ce4894b2 下载对接
658d39ad59946b9eb3cb896ad7b0a2c7e0f407c7 调用安卓ios下载方法
f022a3a68b89844d72d85415835135553ac918a6 生产环境地址
```

git log --graph 可以查看冲突的日记

## 版本回退 上传远程

git log -pretty=oneline

git reset --hard HEAD^ 回退上一个版本

git reset --hard 2321823 回退到指定版本

git push -f origin master 把回退的版本 提交到远程服务器上

git push origin master --force

撤消暂存区的修改（比如你修改了一个 a.txt 文件，然后你添加到暂存区的，这时你想撤消，重新改一下，你可以这样做）

git status 查看一下情况

git reset HEAD a.txt 撤消

git status 再看一下情况

如果你本地删除文件，但是远程仓库还在，你可拉远程到本地（丢弃工作区的修改，从远程拉新的）

git checkout -- test.txt

git checkout -- \* 所有文件

git checkout -- a.txt b.txt

在 test 分支上与 master 分支保持一致（从远程再加一份即可）

git pull origin master

解决冲突（比如 master 分支修改了 a.txt 文件，test 分支修改了 a.txt 文件，如果两个分支合并的话，会发生冲突）

比如：

git checkout test

vim a.txt

git add a.txt

git commit -m "modify a.txt"

git checkout master

vim a.txt

git add a.txt

git commit -m "modify a.txt"

git merge test

提示信息：Auto-merging a.txt

CONFLICT (content): Merge conflict in a.txt

Automatic merge failed; fix conflicts and then commit the result.

这里我们编辑 a.txt 文件,把重复的删除

vim a.txt

```
git add a.txt
git commit -m "modify a.txt"
git push origin master
```

git fetch : 相当于从远程获取最新版本到本地 , 不会自动 merge 比如 :

```
git fetch origin master:tmp
git diff tmp
git merge tmp
```

git pull : 相当于从远程获取最新版本并 merge 到本地 比如 :

```
git pull origin master
```

合并 commit ( 比如把三个 commit 合并 )

```
git log --pretty=oneline
```

```
a7209a68dc44c6fccab649b5bdd2f36a8fdde6ec s3
5c8f255e0a73daaf0ce7436bfe021058adfb78b4 s2
dddae5419216bbe2ccda4c4de2cc4823010012a6 s1
5232a4a7544b15bb5b321f96964b10d5b92024a9 test1 modify
```

```
git rebase -i 5232a4a7544b15bb5b321f96964b10d5b92024a9
```

出现一个 vi 编辑器 , 然后修改

```
pick    a7209a6 s3
s       5c8f255 s2
s      dddae54 s1
```

s:代表合并上一个 commit

然后 wq

然后又出来一个修改 commit 信息的 vi 界面 , 然后修改完成提示信息 ,

提示信息格式 :

modify solug in the homepage

<http://jira.corp.ebates.com/ex-58>

再 wq

```
git log --pretty=oneline  
git checkout master  
git pull origin master
```

```
git checkout ex58  
git rebase master  
git push origin ex58 --force
```

忽略本地修改的文件 ( 不想让本地修改的文件上传远程 , 又不想 git status 时看到 )

- 1、可以编辑 .gitignore 这个文件
- 2、使用命令忽略

```
git update-index --assume-unchanged
```

```
library/Extrabux/PaymentDestinationHelper.php  #忽略跟踪
```

```
git update-index --no-assume-unchanged /path/to/file #恢复跟踪
```

git 切换到远程分支(参考 : <http://www.cnblogs.com/xiaouisme/p/5754656.html>)

1、git branch -a (可以查看本地+远程分支列表)

```
* master
remotes/origin/HEAD -> origin/master
remotes/origin/add-bonus-for-purchase
remotes/origin/cmbc-first-three-cashback
remotes/origin/co-brand-credit-card
remotes/origin/cobrand
remotes/origin/cobrand-di
remotes/origin/exp
remotes/origin/extrabux-softlogin
remotes/origin/incubator
remotes/origin/master
remotes/origin/promotion-transrushcode
```

2、如果想切换到 origin/cmbc-first-three-cashback 分支，我们可以

```
git checkout -t origin/cmbc-first-three-cashback
```

查看某个文件的日记 如：

```
git log --pretty=oneline library/ExtrabuxDev/Model/Purchase.php
```

查看某个 commit 修改的数据

```
git show af8358080c325e16b06650b916227d79ddb497ab
```

把某个文件回滚到线上某个 commit 的状态

```
git checkout af8358080c325e16b06650b916227d79ddb497ab
library/ExtrabuxDev/Model/Purchase.php
```

git 子模块相关

如果 git clone <http://10.211.62.41:82/yimishiji/yii2frame.git> 这个仓库之后，发现有一个文件是.gitmodules 这说明有子模块，然后执行：



来初始化你的本地配置文件，  
`git submodule init`

从那个项目拉取所有数据  
`git submodule update`

如果没有.gitmodules 这个文件，你可以添加子模块

如：相关子模块，中台 api

`git submodule add -f http://10.211.62.41:82/yimishiji/YSCApi.git yscapi`

`git rm -r --cached` 显神威

但是有时候，gitignore 考虑不全面，发现有不该提交的文件已经提交后，仅仅在.gitignore 中加入忽略是不行的。这个时候需要执行：

`git rm -r --cached filename`

去掉已经托管的文件，然后提交即可。

比如：把 index.php 文件移除

`git rm -r --cached index.php`

`git rm -r --cached .` 去掉所有文件

### 3.Git 中的 tag

tag 是什么意思呢，git 有提交记录了，为什么还要 tag 呢？

答：tag 中文是标签的含义，我们可以在 commitId 上打个 tag，这样的好处是，如果我们要上线的话，我们可以只发布某个 tag, 不需要根据 commitId 走，这样的话，我们还可以继续提交我们的 commitId，就可以互不影

响

查看 tag

```
git tag
```

简单的创建一个标签

```
git tag v1.2
```

创建一个含附注类型的标签 -a 表示注释 -m 提示的消息（在某个 commitId 上打一个 tag）

```
git tag -a v1.2 -m "version 1.2"
```

把指定的版本推送到服务器

```
git push origin v1.2
```

一次推送所有本地新增的标签到服务器

```
git push origin --tags
```

git stash 介绍

git stash 会把所有未提交的修改（包括暂存的和非暂存的）都保存起来。

把改动的缓存，并备注信息

```
git stash save "第一个缓存"
```

查看缓存的列表

```
git stash list
```

把指定的索引的修改恢复出来

```
git stash pop --index 0
```

删除某个指定的缓存

```
git stash drop stash@{0}
```

删除 git 的 http 方式密码

```
git config --system --unset credential.helper
```

密码过期

```
git config credential.helper 'cache --timeout=1'
```

```
git remote set-url origin http://10.211.62.41:8081/shinmigo/mms/mms-vue.git
```

```
sencha app watch/build
```