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：

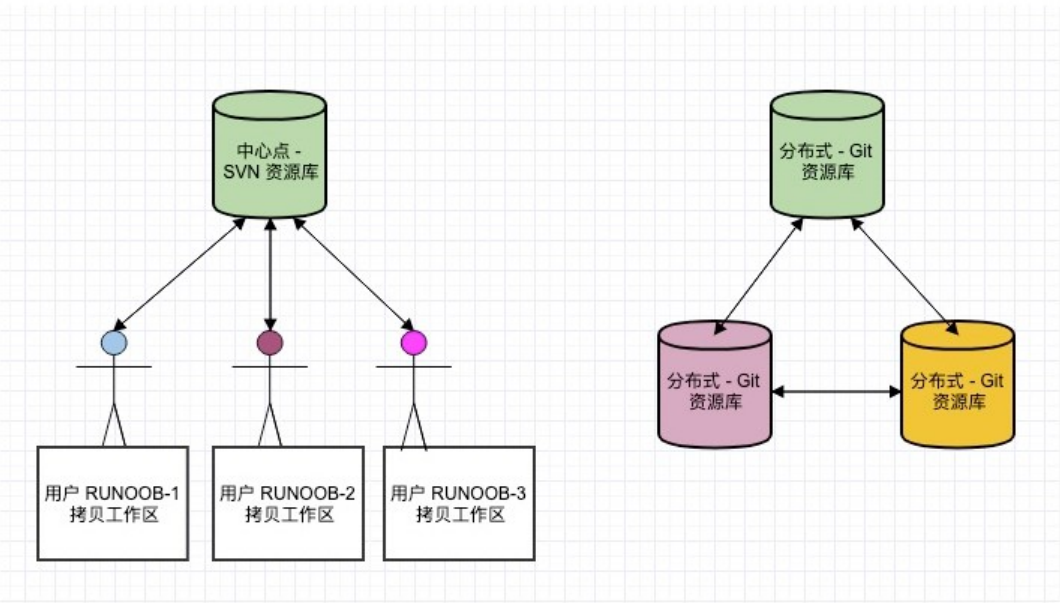
1、Git 是分布式的，SVN 不是：这是 Git 和其它非分布式的版本控制系统，例如 SVN，CVS 等，最核心的区别。

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 orgint/test

查看 git 版三
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

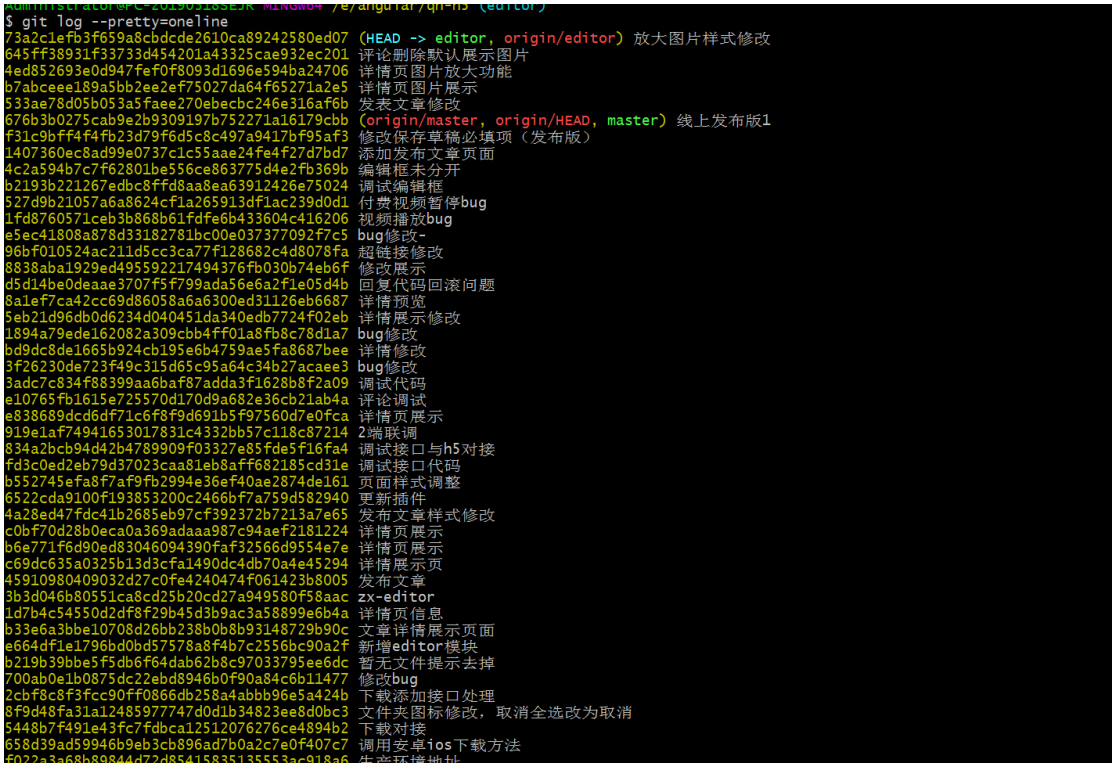
合并分支(切换到 master)
git merge test

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



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：相当于是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：相当于是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