Git help 常用命令 和解释

Git help -a 全部的git命令。

Git help -g git的手册

Git help add 具体怎么用

F 上翻页

B 下翻页

Q推出显示的信息

Git config

安装后需要配置。

3个设置

System

GOLOBAL

项目

Git config --global user.name “人名”

Git config --global user.email “邮箱”

Git config --list 显示信息

清除设置git config --unset --global user.name

设置输出界面的字体颜色

Git config color.ui true

.gitconfig文件中 进行设置。当前用户的主目录下

Cat ~/.gitconfig

Ls 查看文件

删除文件

Git rm index.html

新教程。

Git basic

Git branching

Git workflows

Why git

版本控制。

开发中，很快文件夹会很多，容易混淆，用工具 源代码管理工具，cvs git

Git big keeper linas 自己开发一个，一呼百应。

分布式开发

分支和合并 指时间。

非常灵活的

工具怎么下载安装。 Linux 和 mac 自带

Git 有3个厂商。

Git config --global user.name “人名”

Git config --global user.email “邮箱”

Git config --list 显示信息

Git help 常用命令 和解释

Git help -a 全部的git命令。

Git help -g git的手册

Git help add 具体怎么用

建立版本控制。

Git init 开始进行git 版本控制了。 初始化仓库。

Git add index.html 跟踪某个文件。

Git add \*.html 跟踪所有html文件。

Git add . 跟踪所有。

Git status git 的状态。

处于暂存区

Git commit 只会提交暂存区的文件。

Git log 会有条记录。

还可以追加内容

Echo “new content” >>README.md

Git 有三个区 工作区 、暂存区、历史提交区

文字变红了 说明工作区改变，放到暂存区需要git add README.md,通过git status 来看他的状态,在提交git commit.

可以自定义提交时候的文字。Git commit -m “您已经提交”。 M,message.

直接提交 省略中间放到咱去区域，git commit -a ，前提是之前放过到暂存区，提交过。

Git diff 具体文件名/. (所有文件名) 查看改变了哪里？

Git diff --staged

Git rm 不要乱用

Git mv 旧名 新名。

1.工作区回退 还没有放到暂存区。 Git checkout --readme

2．若果 git add readme 了 即 放到暂存区域了 ，需要把它拉回到工作区 ，git reset HEAD readme .

3.如果已经提交了，git reset --hard 前一步提交的串值（git log 上次的commit值）

Cat 文件名 查看文件的内容。

Git checkout --file.回到

Git stash is better.

Git stash apply 用垃圾桶里的东西。

Git tag 某个节点 哪个点

Git tag -a v1.0 -m “my vision 1.0” 打了个标签

再git v1.0

Git show v1.0

Git tag -a v0.1 -m ”my vision 0.1” commit的串号

Git 可以自动补全

Git 补全

Git alias别名配置 ,就是对一些命令 进行重新定义 ，放到了gitconfig中去了。

Core

Editor =vim 这是个什么编辑器。

这里有作者的gitconfig 文件<http://github.com/dorayo/dotfiles.git>

Git分支 精髓所在。

仓库，git init 之后就

Mkdir branchDemon

建立 3个文件

Echo “readme” >README

Echo “test” >test.rb

Echo “xxx”>LISENCE

Git status

三个文件没跟踪，需要跟踪下。

Git add .

Git commit -m ”inti commit”

Git

**Git 分支创建和切换原理**

自己开发时候也可以用，创建分支 master 代表分支

Cat master 会显示一个HASH串号

分支在哪保存。分支永远记录最新的提交，分支记录最新的HASH串号

Git log 调查记录，看到最新的 commit 串号

List cd.. 进入到.git 中 再进入 cd refs 有heads tags

Cd heads ls 有master cat master 会看到和最新提交的HASH一样。因此master 是最新提交。

创建新分支 echo “master串号” >test 写到test中 这是就有两个分支了

Git br 就能看到有几个分支了，并显示 在哪个分支上

分支很廉价

删除分支 rm -rf test 再 git br 该分支就没了

Pwd 是路径吗？

当有多个分支后

.git 下的head 文件记录了 当前分支master

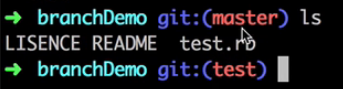
Cat head 查看head文件

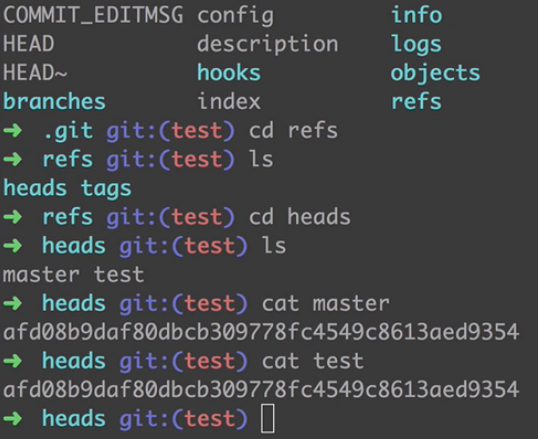
Ref:refs/heads/master

Echo “ref:/heads/test”>head 改成test 改不了

Vi head 打开这个文件直接修改 把

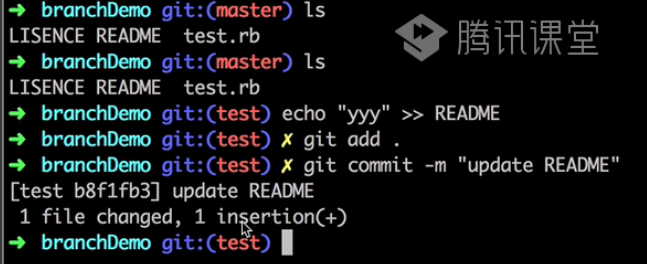
改成 变成

 切换分支 ，因此切换分支的原理 就是 修改head vi head



Refs 中 查看两个分支hash

修改 test echo “yyy”>test



在test 上进行一次提交。此时test 和master hash 已经不一样了。分支的移动

如果两个文件 各自进行 提交 就变成 分叉了 diverged

实际创建和删除分支 是用git 命令来完成 的 ，上述方法 是 hack 用的。

Git branch 列出、创建、删除、分支

git br列出 git br –d test删除 git br test 创建

Git checkout 选择新分支 切换分支 git co master

Git merge 合并

Git cherry-pick 分支个别的 提交 剪切 合并过来。

Git co –b test 切换分支 如果没有 可以创建 -b

先切换到另一个分支，

Git merge test 这样就可以合并了。