**单词：**

untrack 未加入暂存区 modified 未修改 HEAD 指的是当前指针指向

**注意事项：**

git操作提交等命令要进入和.git 同级目录上

//只要执行config --global的命令就会创建一个.gitconfig 的配置文件 这文件的地址就在当前用户文件下这是全局配置

工作区：也就是本地文件的区域

版本库中暂存区：就是使用git add命令之后，本地工作区的文件加到暂存区

版本库当前分支：也就是使用 git commit 之后，暂存区的东西到版本库-当前-分支。在当前分支的操作和所有文件都是存在当前分支里和别的分支没关联（查看也只能切换到当前分支才能看到）

**命令及隐藏事项：**

git config --global user.email “1131534180@qq.com”

git config --global user.name "tancanghai"

git init 创建库

//执行这个会在你的库里新增加一个 .git的配置目录 目录底下有配置文件 这配置文件是隐藏的 这目录底下有config配置文件

git config user.email “1131534180@qq.com”会在你库里创建一个用户

git clone 会在你的项目下生成./git配置文件 https://github.com/tancanghai/CFChain.git 会生成一个CFChain/项目/.git

在.git 同级目录下创建.gitignore 就是忽略不受版本控制配置 .gitignore 只能忽略原来没有被 track 的文件，如果某些文件已经加入了版本管理中，修改 .gitignore 是无效的。解决方法：先把不想记录文件的本地缓存删除，再在 .gitignore 中添加忽略该文件，然后再提交。

warning: LF will be replaced by CRLF in 解决办法：git config --global core.autocrlf false

必须git commit 之后才能产生主分支之外的分支 分支操作要看作用点 作用点之前做的都一样 分支中还可以建立分支

创建新分支：git branch branchName 切换到新分支：git checkout branchName

上面两个命令也可以合成为一个命令：git checkout -b branchName

git clone时其实把所有的分支代码都下载了，只需要本地checkout就可以切换到对应分支，

git clone https:*//github.com/xxx*

git branch -vva *// 会输出所有分支名 但除了当前分支 其他分支都是remote（远程分支）开头的*

git checkout –b 本地分枝名 origin/目标分支名//创建分支并且切换到分支连接远程分支

**命令：**

git init

git add .

git commit -m "tan"

git branch 查看分支；

git branch 分支名 创建分支；

git checkout 分支名 切换分支；

git merge 分支名 合并分支；

git branch --merged 查看已经合并的分支

git branch --no-merged 查看没有合并的分支

git status 命令用于显示工作目录和暂存区的状态

git log 命令显示从最近到最远的提交日志

git reflog 用来记录你的每一次命令：所有命令

git rm --cache a.php 把文件从暂存区撤消 这是第一次 ；之后使用git reset HEAD a.php 内容没改只是撤销从暂存区

git checkout --a.php 把这文件回复到上次加入暂存区之前的操作

使用tag打标签 git tag 常看当前标签； git tag 标签名 打标签 如git tag v1.0

生成zip代码发布压缩包 git archive master --perfix='hdcms/' -forma=zip>hdcms.zip

把当前版本回退到上一个版本，就可以使用git reset命令：git reset --hard HEAD

git reset --hard (commitid) 回到commitid那个版本 commitid可以只填写前几位数字

git remote : git remote add <远程服务地址自己编辑别名 > <url> ;与远程建立连接

例如： git remote add origin <https://github.com/tancanghai/CFChain.git>

git pull <远程主机名> <远程分支名>:<本地分支名>

git push <远程主机名> <本地分支名>:<远程分支名>

git checkout -b dev/1.5.4 origin/dev/1.5.4 # 从本地dev/1.5.4分支取得到远程分支/dev/1.5.4

把远程库的 dev 分支拉倒本地的 dev 分支时使用指令：

$ git checkout -b dev origin/dev

说明：

-b 表示本地创建 dev 分支并切换到 dev 分支上；

origin/dev 表示远程库的 dev 分支。

**远程分支合并**

