git总结

**一、常用命令**

1. 克隆：git clone \*\*
2. 自定义：git clone \*\* my-hello-world
3. 查看本地分支：git branch;
4. 查看所有分支：git branch –a;
5. 创建分支：git branch <branch-name> # 例：git branch develop
6. 切换（检出）分支：git checkout <branch-name># 例：git checkout –b develop
7. 创建并切换到新建分支: git checkout –b <branch-name># 例：git checkout –b develop
8. 跟踪新文件或者已有文件的改动

git add <file-name> # 添加一个文件 例：git add ./index.html;

1. 监控工作区的状态树，把工作时的所有变化提交到暂存区，包括文件内容修改（modified）以及新文件（new），但不包括被删除的文件

git add .

10、监控已经被add的文件（即tracked file）,他会将被修改的文件提交到暂存区。add –u 不会提交新文件（untracked file）。 （git add –update的缩写）

git add -u

11、监控所有的文件，新的未跟踪的，已删除的，已修改的（git add –all的缩写）

git add –A

12、提交代码

git commit –m <comment> # 例：git commit –m “修复了issue-1：错别字”

13、从远端拉取代码

git pull

14、上传代码到远端

git push

15、切换到master主干并将develop分支的代码合并过来

git checkout master

git merge<branch-name> # 例：git merge develop

**二、重要命令**

1. 配置全局用户名

git config --golbal user.name“MRZYD”

1. 配置全局邮箱

Git config –global user.email”[2416389097@qq.com](mailto:2416389097@qq.com)”

1. 显示所有配置列表

git config –list

1. 显示当前目录所有文件包括隐藏文件

ls –la

1. 初始化git仓库

git init

1. 添加该文件到暂存区

git add index.html

1. 查看当前状态

git status

1. 从暂存区移除到工作区

git rm --cached index.html

1. 查看当前状态

git status

10、添加所有的文件到暂存区

git add .

11、查看当前状态

git status

12、将缓存区的内容提交到历史区(此处的"-m"表示不跳到另一个页面，后面跟的"init project"是备注)

git commit . -m "init project"

13、查看当前状态

git status

14、添加所有文件到历史区

git commit .

15、查看当前状态

git status

16、将工作区的修改撤销；取回暂存区的文件

git checkout index.html

17、查看当前状态

git status

18、查看哪些文件被更改过

git status –s

19、查看工作区和暂存区的文件区别

git diff

20、查看该文件被更改过的详细信息

git diff index.html

21、查看该文件被更改过的详细信息

git diff index.js

22、查看该文件被更改过的详细信息

git diff index.css

23、查看提交历史

git log

24、查看当前状态

git status

25、查看工作区和暂存区文件的区别

git diff

26、添加所有文件到暂存区

git add .

27、查看当前状态

git status

28、查看工作区和暂存区文件的区别

git diff

29、查看暂存区和历史区的区别

git diff –cached

30、查看工作区与上一个版本的不同（更改文件后不提交到历史区可用该命令查看）

git diff HEAD

31、查看提交历史

git log

32、查看当前状态

git status

33、提交当前暂存区的文件（"."可写可不写）

git commit

34、查看提交历史

git log

35、提交显示到一行

git log –oneline

36、图形化显示提交的

git log –graph

37、所有文件回到上一个版本

git reset --hard HEAD^

38、查看所有的提交

git reflog

39、回到指定版本

git reset --hard 1b8b6b6

40、查找文件中带有project的文件并显示到一行

git log --oneline --grep="project"

41、暂存区和历史区回到上一个版本，工作区不变

git reset --mixed HEAD^

42、查看当前状态

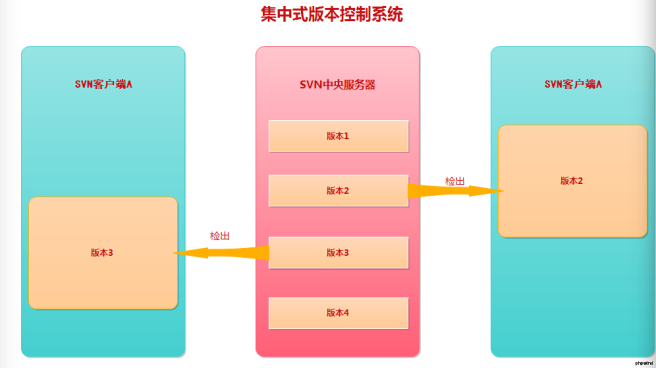
git status

43、通过管道筛选所有js文件

ls | grep \*.js

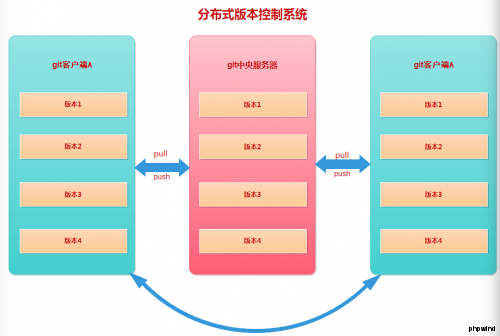
**三、概念**

1、集中式版本控制系统

版本库是集中存放在 **中央服务器** 的，而干活的时候，用的都是自己的电脑，所以要先从中央服务器取得最新的版本，然后开始干活，干完活了，再把自己的活推送给中央服务器。中央服务器就好比是一个图书馆，你要改一本书，必须先从图书馆借出来，然后回到家自己改，改完了，再放回图书馆。

集中式版本控制系统最大的毛病就是必须 **联网** 才能工作，如果在局域网内还好，带宽够大，速度够快，可如果在互联网上，遇到网速慢的话，可能提交一个10M的文件就需要5分钟，这还不得把人给憋死啊。

2、分布式版本控制系统

与集中式版本控制系统有何不同呢？首先，分布式版本控制系统根本没有 **中央服务器** ，每个人的电脑上都是一个 **完整的版本库** ，和集中式版本控制系统相比，分布式版本控制系统的安全性要高很多，因为每个人电脑里都有 **完整的版本库** ，某一个人的电脑坏掉了不要紧，随便从其他人那里复制一个就可以了。而集中式版本控制系统的中央服务器要是出了问题，所有人都没法干活了。 分布式版本控制系统通常也有一台充当“中央服务器”的电脑，但这个服务器的作用仅仅是用来方便“交换”大家的修改，没有它大家也一样干活，只是交换修改不方便而已。

**四、创建版本库**

1、打开命令行

2、创建一个空目录

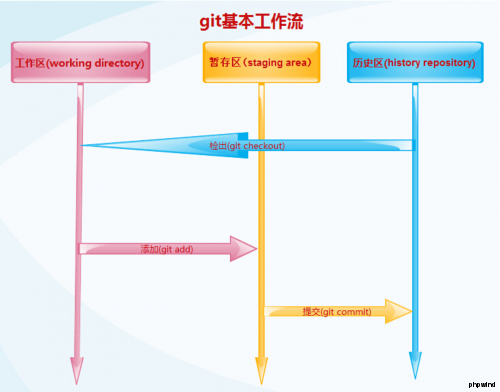
mkdir zyd

3、初始化仓库

git init

**五、工作流**

你的本地仓库由 git 维护的三棵“树型(有层级关系)区域”组成。

* 第一个是你的 **工作目录**，对应你的物理目录,树型结构，也就是编辑代码的地方，写代码的地方；
* 第二个是 **暂存区**，工作区和历史提交的中间缓存，代表需要提交的工作状态，维护虚拟树型结构,临时保存你的改动；
* 最后是 **历史区**，树型结构，历史仓库,指向你最近一次提交后的结果。

大部分时候是先提交到暂存区中，再提交到历史里面。也可以从历史记录里面检出数据到暂存区和工作区。   
.git文件夹中包括暂存区和历史区

**六、添加文件**

**说明**   
echo xxx > yyy.txt 表示把 xxx内容保存到yyy.txt中，你也可以打开记事本进行编辑，手工把xxx写入yyy.txt中并保存   
为了简明起见，我们创建一个 index.html 作为练习：   
$ echo 1 > index.html   // 把1输出到新创建的index.html文件中  > 表示清空并写入   
$ echo 2 >> index.html // 追2加到index.html中 >>表示在原来文件的末尾追加   
$ cat index.html //查看文件内容   
1   
2   
  
一定要保存到 **zyd** 目录下（子目录也行），因为这是一个Git仓库，放到其他地方Git再厉害也找不到这个文件。   
用命令git add告诉Git，把文件添加到 **仓库中的暂存区** ：   
  
$ git add index.html //把文件添加到暂存区   
warning: LF will be replaced by CRLF in index.html.   
The file will have its original line endings in your working directory.   
//警告: index.html 中的 /n将会被/r/n替换掉，在原始目录中文件将保留最原始的换行符   
//这个设置跟git-scm安装时的设置有关，因为在window下换行是\r\n,在linux下是\n   
  
git add 实际上是个脚本命令，没有任何显示，说明添加成功。

**七、提交文件**

**说明**   
用命令git commit告诉Git，把文件提交到仓库：   
  
$ git commit index.html -m"在index.html中增加了 1 2"   
  
[master (root-commit) da19241] 在index.html中增加了 1 2  // da19241是这次提交的 commit ID,而且是在master分支上提交的   
  
 1 file changed, 2 insertions(+)  //增加了个文件，插入2行   
 create mode 100644 index.html  //创建的文件权限模式是 比如一个文件是644，那么8 进制=100644 数字含义请参考 [linux权限位]

(http://www.php100.com/html/program/linux/2013/0904/2490.html)

简单解释一下git commit命令，-m后面输入的是本次提交的说明，可以输入任意内容，当然最好是有意义的，这样你就能从历史记录里方便地找到改动记录。   
 如果你没有添加-m参数的话表示会弹出一个编辑页面，可以输入你的注释然后按esc退出编辑模式，再输入:wq!退出此编辑器。

