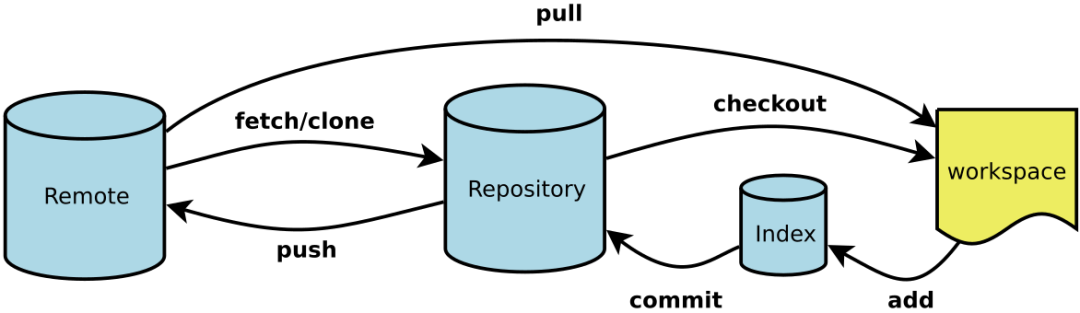
首先看下工作流程



区域划分：

远程仓库

本地仓库

缓存区

工作区

1. 假设先有了本地仓库：

在某个特定的目录下，

git init

然后增加我们的文件后，

Git add ./

Git commit –m “注释”

这个时候文件提交到本地仓库

同样的如果修改了，我们仍然需要add, 然后commit.

然后绑定本地仓库到远程仓库：

Git remote add origin <https://github.com/kevin-jk/Hello-world.git>

其中origin是远程库的名字，可以改成其他的名字，不过一般是origin.

然后下一步就是推送

Git push –u origin master

其为git push –set-upstream origin xxx的简写

2 直接clone远程仓库

大多数时候，我们已经有了远程仓库，此时首先要做的就是clone远程仓库：

git clone <https://github.com/kevin-jk/Hello-world.git>

此时Git 的 clone 命令会为你自动将其命名为 origin，拉取它的所有数据，创建一个指向它的 master 分支的指针，并且在本地将其命名为 origin/master。 Git 也会给你一个与 origin 的 master 分支在指向同一个地方的本地 master 分支。

实际上我们一般情况下，不会直接在master分支上操作。 此时我们需要检出对应的远端分支。

可以先看下有哪些分支：

git branch –r

然后

git checkout -b dev\_20190616-02 origin/dev\_20190616-02

此时git会自动的跟踪这个分支。

另外，也可以单独创建分支。

一些常用的命令：

Git DIFF:

 git diff ： 对比工作区(未 git add)和暂存区(git add 之后)

 git diff --cached: 对比暂存区(git add 之后)和版本库(git commit 之后)

 git diff HEAD: 对比工作区(未 git add)和版本库(git commit 之后)

那么如果我们发现已经修改的文件需要回退，应该如何操作呢？

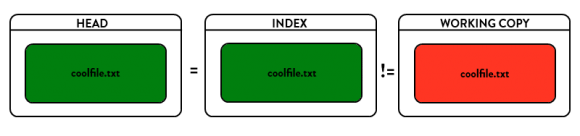
1. 只是修改了工作区的文件， 还未add： git checkout . （git checkout -- filename）
2. 已经add: git reset --hard
3. 已经commit : git reset –hard xxversion (例如：HEAD^)

Reset的一些参数：

**1. --soft**

仅仅移动当前Head指针，不会改变工作区和暂存区的内容    


**2.  --mixed**

是reset的默认参数,移动head指针，改变暂存区内容，但不会改变工作区   


**3.  --hard**

当前head指针、工作区和暂存区内容全部改变    
