git(自己的版本)

用户设置

- 安装完成后，还需要最后一步设置，在命令行输入：

```
git config --global user.name "zfpx"
git config --global user.email "zfpx@126.com"
```

> - 因为Git是分布式版本控制系统，所以每个机器都必须自报家门：你的名字和Email地址。
> - 注意git config命令的–global参数，用了这个参数，表示你这台机器上所有的Git仓库都会使用这个配置，当然也可以对某个仓库指定不同的用户名和Email地址。

右键选择Git bash(安装git-scm后会有此选项)打开命令行.

创建一个空目录

```
mkdir zfpx //创建一个文件夹zfpx
cd zfpx  //进入zfpx
pwd
//pwd命令用于显示当前目录。在环境中这个仓库的位置
```

初始化仓库

```
git init
ls-al
```

> 注释：
> **git init** //初始化:安装了一个.git
> **ls -al**如果你没有看到**.git**目录，那是因为这个目录默认是隐藏的，用ls -al命令就可以看见。

在work中创建一个写好内容1 2的index.html文件

```
$ 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
```

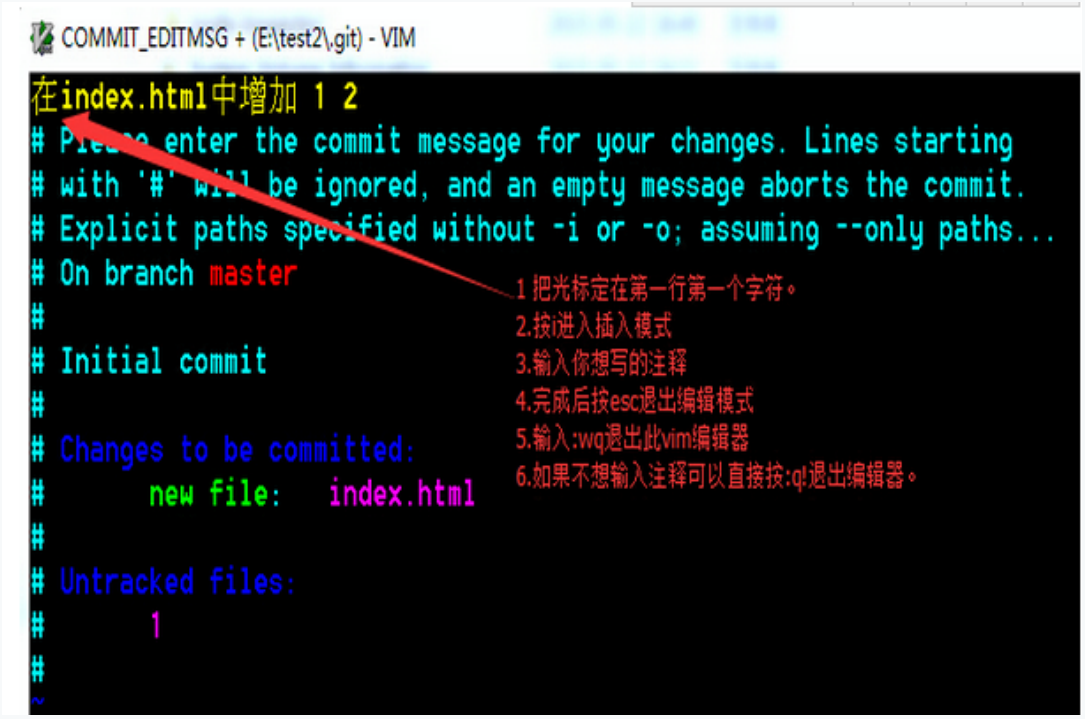
```
rm index.html  //删除index.html
```

- 代码中的空格一定不能丢
- 一定要保存到 zfpxgit 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。

把文件添加到暂存区并且推送到仓库

```
git add index.html
//git add .
git commit -m'注释（改动了什么）'
```

- 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。



- git commit命令执行成功后会告诉你，1个文件被改动（index.html文件），插入了两行内容（index.html有两行内容）。
- 为什么Git添加文件需要add，commit一共两步呢？因为commit可以一次提交很多文件，所以你可以多次add不同的文件，比如：

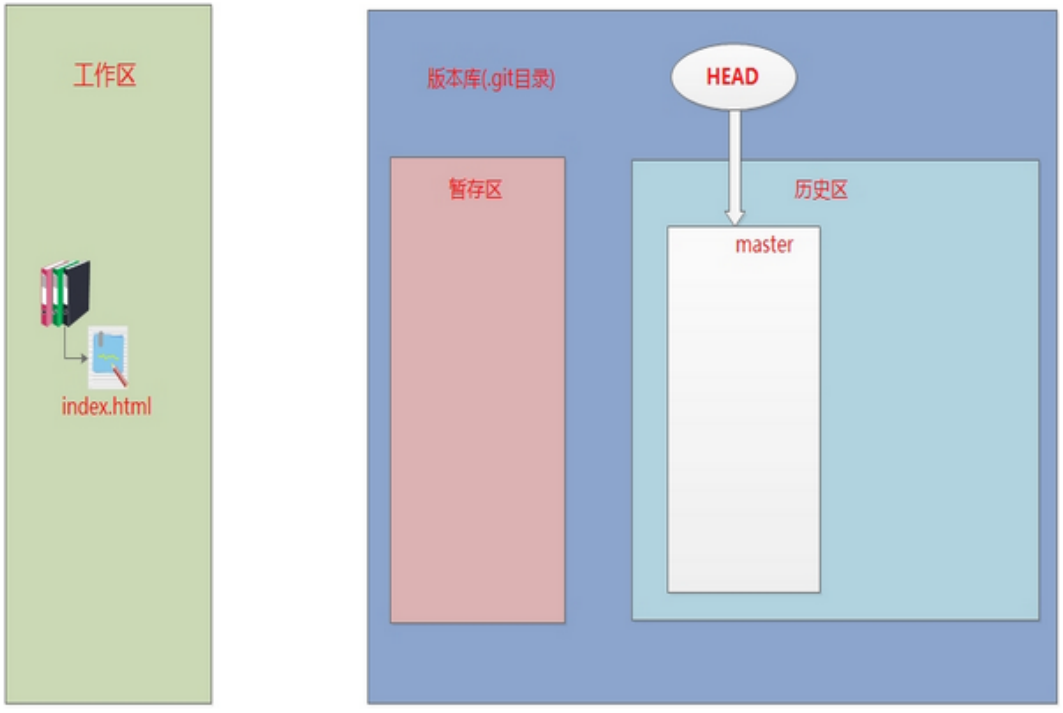
```
$ git add file1.txt
$ git add file2.txt file3.txt
$ git commit -m "add 3 files."
```

工作区（Working Directory）

- 就是你在电脑里能看到的目录，比如我的zfpxgit文件夹就是一个工作区：

版本库（Repository）

- 工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库。 Git的版本库里存了很多东西，其中最重要的就是称为stage（或者叫index）的暂存区， 还有Git为我们自动创建的第一个分支master，以及指向master的一个指针叫HEAD。



分支和HEAD的概念我们以后再讲。

补充：

前面讲了我们把文件往Git版本库里添加的时候，是分两步执行的：

- 第一步是用git add把文件添加进去，实际上就是把文件修改添加到- 暂存区；
- 第二步是用git commit提交更改，实际上就是把暂存区的所有内容提交到当前分支。

- 因为我们创建Git版本库时，Git自动为我们创建了唯一一个master分支，所以，现在，git commit就是往master分支上提交更改。
- 简单理解为：
  需要提交的文件修改通通放到暂存区，然后，一次性提交暂存区的所有修改。

练习

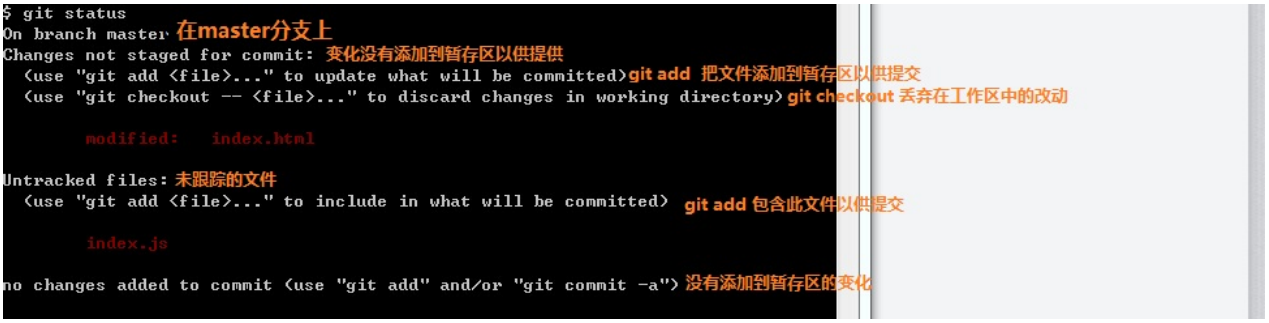
> 1、先对index.html做个修改，加一行内容（第一次添加了个王伟），又加了个（3）
>
> - 为什么加文字会是乱码

```
Administrator@N7ALX5DWW6HPZRD /E/zfpxgit (master)
$ cat index.html
1
2
乱码
3
```

> 2、在工作区新增一个index.js文本文件（内容随便写）。

添加后查看状态：

```
git status
```



```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        index.js

no changes added to commit (use "git add" and/or "git commit -a")
```

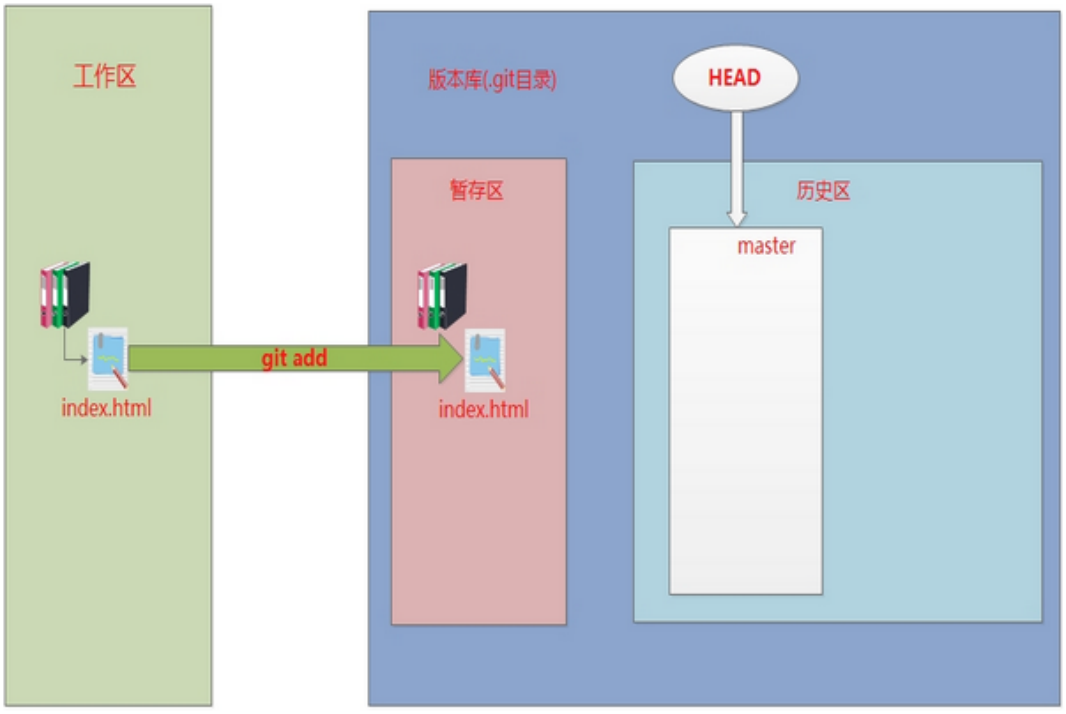
> - 注：工作区：**index.js和index.html 是红色**的
> - index.html被修改了，而index.js还从来没有被添加过，所以它的状态是Untracked。

现在，使用两次命令git add，把readme.txt和index.js都添加后，用git status再查看一下：

```
git add index.html
git add index.js
```

- 现在这样子：

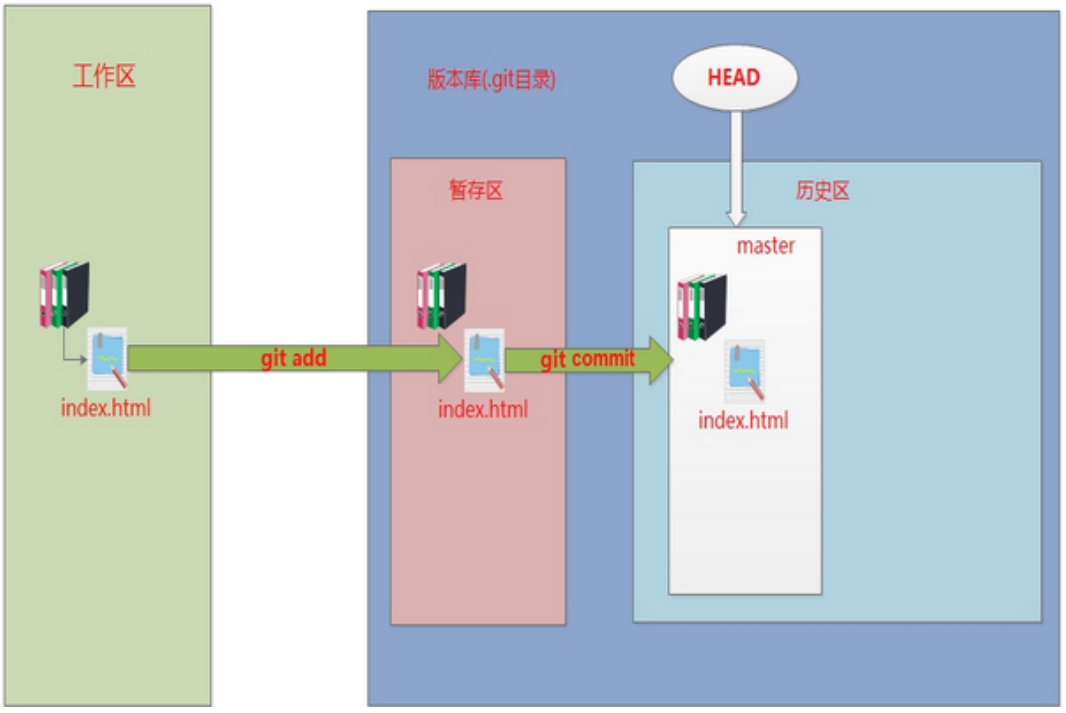
现在，暂存区的状态就变成这样了：



- 所以，git add命令实际上就是把要提交的所有修改放到暂存区（Stage），然后，执行git commit就可以一次性把暂存区的所有修改提交到分支。
- 一旦提交后，如果你又没有对工作区做任何修改，那么工作区就是“干净”的：

现在版本库变成了这样，工作区、暂存区和历史区完全一致：

现在版本库变成了这样，工作区、暂存区和历史区完全一致：



代码对比

运行git status命令看看结果：

```
git status  //查看状态
```

```
git status -s
```

- 工作区和暂存区

```
git diff
```

```
王伟@pc123456 MINGW64 ~/Desktop/交作业/work (master)
$ git diff
diff --git a/index.html b/index.html
index e49a1b8..cab82e9 100644
--- a/index.html
+++ b/index.html
@@ -2,4 +2,5 @@
 2

 3
-4
\ No newline at end of file
+4
+5
\ No newline at end of file
warning: LF will be replaced by CRLF in index.html.
The file will have its original line endings in your working directory.
```

> 工作区 ：红色
> 暂存区 ：绿色

- 工作区和暂存区

```
git diff
```

- 工作区和历史区

```
git diff 分支名（此时分支名是什么）
```

- 暂存区和历史区

```
git diff --cached
```

- 上传之后看历史

```
git log //查看版本（也就是每次改变后的版本）
```

```
王伟@pc123456 MINGW64 ~/Desktop/交作业/work (master)
$ git log
commit 482f781af5371b8be33f8bb49381ce14a782e257
Author: wangwei2017-1 <1129942397@qq.com>
Date:   Mon May 1 21:24:12 2017 +0800

    add 5

commit 540361fcf02a2d9e97423c4a2406ad0b349e7252
Author: wangwei2017-1 <1129942397@qq.com>
Date:   Mon May 1 21:04:34 2017 +0800

    add diff

commit 8b785c1ec5a29725a75dacaa82520f6bb90a9be5
Author: wangwei2017-1 <1129942397@qq.com>
Date:   Mon May 1 20:59:19 2017 +0800

    index.js  index.html

王伟@pc123456 MINGW64 ~/Desktop/交作业/work (master)
$
```

> - 上面哪一个是版本号:commit (前7位数)

- 查看所有历史

```
git reflog
```

- 回到上一个版本

```
git reset --hard HEAD^
```

- 回到上上一个版本

```
git reset --hard HEAD^^
```

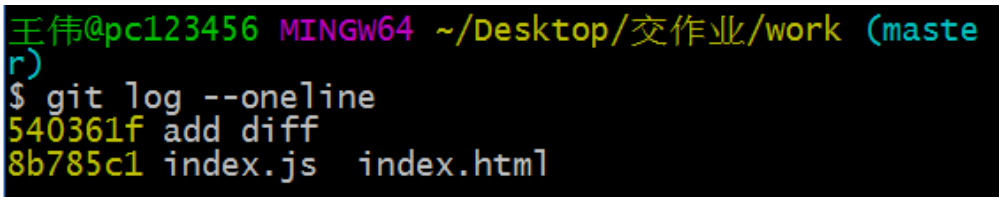
- 回滚某个版本

```
git reset --hard 版本号（版本号是什么）
```

> 版本1：在index.html中增加了 1 2
> 版本2：add index.js
> 版本3：add diff

- 获取版本号

```
git log --oneline
```



$ git log –oneline
521cb3d add diff
728bab9 add index.js

> add 前面的是版本号

- 搜索

```
git log --grep=hello
git log --author=wake
```

分支问题

- 查看当前项目下的分支

```
git branch    //查看当前分支
```

> 默认分支是 master

- 创建分支

```
git branch dev
```

> 可以创建多个分支，但是都是由主分支master复制过来的，要在master基础上修改

- 切换分支

```
git checkout dev
```

- 删除分支

```
git branch -D dev
```

- 创建并切换分支

```
git checkout -b dev
```

- 合并分支

```
git merge 分支名
```

> 默认master是主干，用主干和并分支
> 合并分支后 被合并的分支需要删除
> 合并完成之后，将合并后的文件再次上传

处理冲突

由于两个分支改变了相同的文件，但是内容不同这时候要手动处理，再次提交，就可以完成合并

远程（remote）

github注册之后需要配置邮箱

创建有内容的文件

```
echo 内容 > 文件名
echo 内容 >> 文件名 (追加内容)
```

关联仓库

```
git remote add 名字 地址
```

推送到远

```
-u upstream 你设置后下次推送时可以使用简写
git push origin master -u
git push
```

忽略文件

- .gitignore
  要忽略的文件需要在.gitignore建立之后再add.

> git不能提交空文件夹

在github上发布静态网站

- 必须当前项目下建立一个gh-pages的分支
- 将我们需要发布的内容推送到gh-pages这个分支上
- 推送到远程仓库上即可
- github会给你一个在线地址

```
git checkout -b gh-pages
touch ind ex.html
git add .
git commit -m 'add index.html'
git push origin gh-pages
在settings中可查找到网址+ 文件名即可 (默认会展示index.html)
```

将文章发送给老师（通过github）

- 老师 https://github.com/zhufengzhufeng/201701node_homework.git
- 组长
  组长fork（叉子）老师代码，你的代码和我的一样，我改变了代码不会自动同步你的项目
- 组员
  组员fork组长的仓库

获取自己仓库的代码

```
git clone https://github.com/wakeupmypig/201701node_homework.git le
ader
```

组长克隆自己的代码

- 1.组长建立文件夹
- 2.推到自己的仓库上

组员克隆自己的代码

- 3.添加组长为远程仓库地址

```
git remote add leader 地址
```

- 4.拉取组长最新代码

```
git pull leader master
```

- 5.找到自己组建立自己的文件夹，放入自己的作业
- 6.提交到自己仓库上
- 7.发送pull request
- 8.组长合并

1.

- 组长fork老师
- 组员fork组长

2.

- 组长克隆自己代码
- 组员也克隆自己的代码

3.

- 组长建立自己组的文件夹，里面放入组员信息
- 组长提交到自己仓库

4.

- 组员和组长建立联系

```
git remote add leader 组长的地址
```

- 组员拉取组长最新代码

```
git pull leader master
```

- 将自己写的内容放到文件夹中
- 组员将内容提交到自己的仓库上
- 提交后发送pull request

5.

- 组长合并代码

组员要提交作业?

- 先获取组长的最新代码
- 放入自己的文件提交到自己的仓库上
- 发pull request请求

组长怎么提交给老师?

- 获取自己最新代码
- 获取老师最新代码
- 放入自己的文件，推送到自己的仓库上
- 发pull request请求