## git init

使用 `git init` 可以在当前目录下创建一个本地 git 仓库。命令执行完毕后会产生一个 `.git` 文件夹，用于表示该文件夹是一个 git repository 。通常情况下不要修改这里面的内容。

```
git init
```

## git status

该命令用来查看git 仓库的状态。

```
git status
```

## git add

使用该命令可以将当前的workplace中的文件提交到暂存区中（index）。

```
touch a.txt
git add a.txt
```

上述命令创建了一格 `a.txt` 的文件，同时还将其添加到了暂存区中。

同时执行 `git status` ，此时显示

```
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   a.txt

```

## git commit -m "msg"

在上面的 `git add` 命令后使用了 `git status` 。显示的内容提醒我们有一些变化我们需要来提交。

```
$ git commit -m "add a.txt file"
[master (root-commit) a709746] add a.txt file
 1 file changed, 1 insertion(+)
 create mode 100644 a.txt
```

当我们执行 `git commit` 后，命令符提示我们已经提交了a.txt文件。接下来再次使用 `git status` 来查看一下状态。

```
$ git status
On branch master
nothing to commit, working tree clean
```

## git diff

该指令可以用来查看文件被修改的部分。

我们首先使用文本编辑器来修改上面创建的 `a.txt` 文件，其内容如下：

```
this is not a test.
another line.
```

接下来使用 `git diff` 来查看文件被修改的信息。

```
@@ -1 +1,2 @@
-this is a test.
+this is not a test.
+another line.
```

其中以 `-` 号开头的表示被删除（修改）的行，以 `+` 号开头的表示新增加的行。

接下来，我们将其提交到本地的库中。

```
$ git commit -m "modified a.txt"
[master d3586c5] modified a.txt
 1 file changed, 2 insertions(+), 1 deletion(-)
```

> 注意提交前不要忘记了 `git add a.txt`

## git log

使用该指令可以查看你的提交记录。

```
$ git log
commit d3586c5f55110bb8cba6f9e0afc92feaf2e0c668 (HEAD -> master)
Author: zsstrike <2439758713@qq.com>
Date:   Sun Mar 10 16:29:49 2019 +0800

    modified a.txt

commit a709746dab4fd3a2d8cc442207cca8b4378c91f7
Author: zsstrike <2439758713@qq.com>
Date:   Sun Mar 10 16:14:18 2019 +0800
    add a.txt file
```

## git reset --hard HEAD^

还记得吗？我们的 `a.txt` 里面的内容第一次是

```
this is a test.
```

第二次提交的内容是

```
this is not a test.
another line.
```

如果现在我想把 a.txt 回退到上一个版本，我们可以使用该指令。

```
$ git reset --hard HEAD^
HEAD is now at a709746 add a.txt file
```

现在我们来想查看一下a.txt 里面的内容，看看是否回退到上一个版本了。

```
this is a test.
```

可以发现此时成功回退到上一个版本了。再次执行 `git log`

```
$ git log
commit a709746dab4fd3a2d8cc442207cca8b4378c91f7 (HEAD -> master)
Author: zsstrike <2439758713@qq.com>
Date:   Sun Mar 10 16:14:18 2019 +0800

    add a.txt file
```

> HEAD^ 表示回退到上一个版本，HEAD^^ 表示回退到上上个版本，依次类推。
>
> HEAD~10表示回退10个版本。
>
> 也可以将HEAD换成相应的版本号。

## git reflog

可以查看自己每一次版本更替的记录。

```
$ git reflog
a709746 (HEAD -> master) HEAD@{0}: reset: moving to HEAD^
d3586c5 HEAD@{1}: commit: modified a.txt
a709746 (HEAD -> master) HEAD@{2}: commit (initial): add a.txt file
```

可以发现我们的每次提交记录或者是版本回退记录都会被记录在这里面。

现在我们可以将版本修改为 `d3586c5`

```
$ git reset --hard d3586c5
HEAD is now at d3586c5 modified a.txt
```

重新查看a.txt 的内容，发现又回到最新的版本了。

```
this is not a test.
another line.
```

## git rm

用于删除我们的文件。

接下来我们创建一个b.txt文件并且将其提交到版本库中。

```
touch b.txt
git add b.txt
git commit -m "commit b.txt"
```

接下来我们来删除b.txt 。

```
rm b.txt
git rm b.txt
git commit -m "delete b.txt"
[master 0af6719] commit b.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 b.txt
```

用 `git reflog` 来查看一下我们的操作记录

```
git reflog
e25184a (HEAD -> master) HEAD@{0}: commit: delete b.txt
0af6719 HEAD@{1}: commit: commit b.txt
d3586c5 HEAD@{2}: reset: moving to d3586c5
a709746 HEAD@{3}: reset: moving to HEAD^
d3586c5 HEAD@{4}: commit: modified a.txt
a709746 HEAD@{5}: commit (initial): add a.txt file
```

这样， `b.txt` 就已经从我们的仓库中移除了。

## git remote add origin & git push

现在我们已经基本熟悉了本地库的基本操作方式了。并且我们也已经有了一个本地库，接下来演示如何将本地库关联到远程库。这一段代码可以在GitHub上创建完成后会提示。

```
git remote add origin https://github.com/zsStrike/GitTest.git
git push -u origin master
```

接下来刷新浏览器，你会发现你自己的文件已经上传到了服务器。

## git clone

上面我们演示的是如何将本地库和远程库关联起来，接下来可以从远程库克隆下来我们需要的仓库。

```
git clone https://github.com/zsStrike/GitTest.git
Cloning into 'GitTest'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 9 (delta 0), reused 9 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), done.
```

查看本地的文件夹，可以发现远程仓库的内容已经被克隆到本地了。

## git branch

该命令用来创建分支。接下来我们将自己的工作目录切换到最开始的GitTest目录，然后在该目录下面创建一个 `dev` 的分支。

```
git branch dev
```

`git branch` 不带参数可以查看当前的分支情况。

## git checkout

该命令用于切换相应的分支。我们来切换到刚刚创建的 `dev` 目录中。

```
git checkout dev
Switched to branch 'dev'
```

接下来我们使用 `git branch` 来查看一下分支。

```
git branch
* dev
  master
```

`*` 表示我们目前正处在的目录。

> 为了简单我们可以使用 `git checkout -b dev` 。该条命令的作用是创建并且切换到新的分支中。

接下来我们就可以在这个分区里面进行我们的工作了。

创建一个 `c.txt` 文件，然后再里面写上相应的内容,接着将其提交到本地库，然后切换回原来的 `master` 分支。

```
touch c.txt
git add c.txt
git commit -m "test for dev branch"
git checkout matser
```

## git merge

该命令用来合并指定的分支到当前的分支。我们来将 `dev` 分支合并到 `master` 分支。

```
git merge dev
Updating 0af6719..fbebda2
Fast-forward
 c.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 c.txt

```

接着查看当前的目录，我们可以发现已经多出来了一个 `c.txt` 文件。

```
ls
```

当我们合并完成后，我们可以通过 `git branch -d dev` 来删除dev分支。

```
git branch -d dev
```

# 冲突处理

接下来我们创建一个 `feature1` 的分支并且创建爱你一个文件 `readme.md` ，并且在里面写上'feature1'文本，然后提交它到本地的仓库中。然后再切换回我们 `master` 分支，也创建一个文件 `readme.md` ，并在里面写上 'master' 文本，然后我们尝试着将feature1分支合并到master上面。

```
git checkout -b feature1
touch readme.md
git add readme.md
git commit -m 'readme.md in feature1'
git checkout master
touch readme.md
git add readme.md
git commit -m 'readme.md in master'
```

```
git merge feature1
Auto-merging readme.md
CONFLICT (add/add): Merge conflict in readme.md
Automatic merge failed; fix conflicts and then commit the result.
```

当我们使用 `git status` 时，他会提示我们需要解决冲突。

```
git merge feature1
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both added:      readme.md
```

我们直接查看readme.md文件的内容：

```
<<<<<<< HEAD
master.
=======
feature1.
>>>>>>> feature1
```

我们将这里面的内容修改为

```
master.
feature1.
```

然后再次提交

```
git add readme.md
git commit -m 'conflict fixed'
```

这样我们就成功的解决掉冲突了。

# 多人协作

1. 推送分支

   推送分支直接可以使用 `git push origin <branchName>` ，表示我们希望将当前的分支的内容提交到远程仓库的 `<branchName>` 分支中。

2. 抓取分支

   如果你的团队里面有一个人创建了一个c.txt文件，然后将其提交到了远程库中。与此同时，你也创建了一个c.txt文件，也准备将其提交到远程仓库中。但是提交的时候出现了冲突。

```
git push origin master
To https://github.com/zsStrike/GitTest.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/zsStrike/GitTest.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

按照提示，你需要先将远程库 `pull` 下来。

```
git pull
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 12 (delta 4), reused 12 (delta 4), pack-reused 0
Unpacking objects: 100% (12/12), done.
From https://github.com/zsStrike/GitTest
   0af6719..6af9031  master     -> origin/master
Auto-merging c.txt
CONFLICT (add/add): Merge conflict in c.txt
Automatic merge failed; fix conflicts and then commit the result.
```

然后提示你 `merge conflict in c.txt` 。这个时候你就需要处理冲突了，这和上面的方法是相同的。然后再次提交到远程仓库中。

## .gitignore

这是一个文件，注意没有后缀名。该文件的作用是自定义提交的时候应该忽略掉哪些文件。