# git操作详解

# 实验场景(1)：仓库创建与提交

## 任务

- R0：在进行每次git操作之前，随时查看工作区、暂存区、 git仓库的状态，确认项目里的各文件当前处于什么状态
- R1：本地初始化一个git仓库，将自己在Lab1中所创建项目的全部源文件加入进去，纳入git管理
- R2：提交；手工对提交的部分文件进行修改
- R3：查看上次提交之后都有哪些文件修改、具体修改内容是什么
- R4：重新提交；再次对部分文件进行修改
- R5：重新提
- R6：把最后一次提交撤销
- R7：查询该项目的全部提交记录
- R8：在GitHub上创建名为“ Lab1-学号”的仓库，并在本地仓库建立相应的远程仓库
- R9：将之前各步骤得到的本地仓库全部内容推送到GitHub的仓库中

## 步骤

以下是完成上述步骤的详细Git命令及解释：

R0：查看工作区、暂存区、Git仓库的状态

```
git status
```

解释：`git status`命令可以查看当前工作区、暂存区和Git仓库的状态，显示哪些文件有修改、哪些文件已经暂存、哪些文件还未被追踪等。

R1：本地初始化一个Git仓库，将Lab1中的全部源文件加入Git管理

```
git init
git add .
```

解释：`git init`命令初始化一个新的Git仓库，将当前目录变成Git仓库。`git add .`命令将当前目录下的所有文件添加到暂存区。

R2：提交

```
git commit -m "Initial commit"
```

解释：`git commit -m "Initial commit"`命令将暂存区的文件提交到Git仓库，并添加提交信息"Initial commit"。

手工对提交的部分文件进行修改后，可以使用以下命令进行操作：

R3：查看上次提交之后的文件修改

```
git diff
```

解释：`git diff`命令可以查看工作区和暂存区的文件差异，显示具体的修改内容。

R4：重新提交

```
git add .
git commit -m "Second commit"
```

解释：首先使用`git add .`命令将修改后的文件添加到暂存区，然后使用`git commit -m "Second commit"`命令重新提交，添加提交信息"Second commit"。

再次对部分文件进行修改后，可以使用以下命令进行操作：

R5：重新提交

```
git add .
git commit -m "Third commit"
```

解释：同样，使用`git add .`命令将修改后的文件添加到暂存区，然后使用`git commit -m "Third commit"`命令重新提交，添加提交信息"Third commit"。

R6：撤销最后一次提交

```
git reset HEAD^
```

解释：`git reset HEAD^`命令可以将最后一次提交撤销，将提交的文件回退到暂存区。

R7：查询项目的全部提交记录

```
git log
```

解释：`git log`命令可以查看项目的全部提交记录，显示每次提交的作者、日期、提交信息等信息。

R8：在GitHub上创建名为“Lab1-学号”的仓库，并在本地仓库建立相应的远程仓库

1. 在GitHub上创建名为“Lab1-学号”的仓库，并获取仓库的URL。
2. 在本地仓库中添加远程仓库并命名为"origin"，将GitHub仓库的URL添加为远程仓库。

```
git remote add origin <GitHub仓库URL>
```

解释：`git remote add origin <GitHub仓库URL>`命令将GitHub仓库的URL添加为本地仓库的远程仓库，并命名为"origin"。

R9：将本地仓库全部内容推送到GitHub的仓库中

```
git push -u origin master
```

解释：`git push -u origin master`命令将本地仓库的内容推送到GitHub的仓库中。其中，"origin"是远程仓库的名称，"master"是要推送的分支名。第一次推送时需要使用`-u`参数来将本地的master分支与远程的master分支关联起来。

# 实验场景(2)：分支管理

## 任务

- R1：获得本地Lab1仓库的全部分支，切换至分支master
- R2：在master基础上建立两个分支B1、 B2
- R3：在B2分支基础上创建一个新分支C4
- R4：在C4上，对2个文件进行修改并提交
- R5：在B1分支上对同样的2个文件做不同修改并提交
- R6：将C4合并到B1分支，若有冲突，手工消解
- R7：在B2分支上对2个文件做修改并提交
- R8：查看目前哪些分支已经合并、哪些分支尚未合并
- R9：将已经合并的分支删除，将尚未合并的分支合并到一个新分支上，分支名字为你的学号
- R10：将本地以你的学号命名的分支推送到GitHub上自己的仓库内
- R11：查看完整的版本变迁树
- R12：在Github上以web页面的方式查看你的Lab1仓库的当前状态

# 步骤

以下是完成R1到R12步骤的详细Git命令及解释：

R1：获得本地Lab1仓库的全部分支，切换至分支master

```
git branch -a
git checkout master
```

解释：`git branch -a`命令可以查看本地和远程所有分支的列表。`git checkout master`命令可以切换到master分支。

R2：在master基础上建立两个分支B1、B2

```
git checkout -b B1
git checkout master
git checkout -b B2
```

解释：`git checkout -b B1`命令在master分支基础上创建名为B1的新分支，并切换到该分支。类似地，通过 `git checkout -b B2`命令在master分支基础上创建名为B2的新分支。

R3：在B2分支基础上创建一个新分支C4

```
git checkout B2
git checkout -b C4
```

解释：首先通过 `git checkout B2`切换到B2分支，然后通过 `git checkout -b C4`在B2分支基础上创建名为C4的新分支。

R4：在C4上，对2个文件进行修改并提交

```
git checkout C4
# 对文件进行修改
git add .
git commit -m "Modify files in C4 branch"
```

解释：首先通过 `git checkout C4`切换到C4分支，然后对两个文件进行修改并提交。

R5：在B1分支上对同样的2个文件做不同修改并提交

```
git checkout B1
# 对文件进行不同修改
git add .
git commit -m "Modify files in B1 branch"
```

解释：首先通过 `git checkout B1`切换到B1分支，然后对两个文件进行不同的修改并提交。

R6：将C4合并到B1分支，若有冲突，手工消解

```
git checkout B1
git merge C4
# 若有冲突，手工解决冲突后再提交
```

解释：首先通过 `git checkout B1`切换到B1分支，然后通过 `git merge C4`将C4分支合并到B1分支。如果有冲突，需要手工解决冲突后再提交。

R7：在B2分支上对2个文件做修改并提交

```
git checkout B2
# 对文件进行修改
```

```
git add .
git commit -m "Modify files in B2 branch"
```

解释：首先通过 `git checkout B2`切换到B2分支，然后对两个文件进行修改并提交。

R8：查看目前哪些分支已经合并、哪些分支尚未合并

```
git branch --merged
git branch --no-merged
```

解释：`git branch --merged`命令可以查看已经合并到当前分支的分支列表。`git branch --no-merged`命令可以查看尚未合并到当前分支的分支列表。

R9：将已经合并的分支删除，将尚未合并的分支合并到一个新分支上，分支名字为你的学号

```
git branch -d <branch_name> # 删除已合并分支
git checkout -b <学号> <branch_name> # 创建新分支并合并尚未合并的分支
```

解释：首先使用 `git branch -d <branch_name>`命令删除已经合并的分支。然后使用 `git checkout -b <学号> <branch_name>`命令创建名为学号的新分支，并将尚未合并的分支合并到该新分支。

R10：将本地以你的学号命名的分支推送到GitHub上自己的仓库内

```
git push origin <学号>
```

解释：`git push origin <学号>`命令将本地以学号命名的分支推送到GitHub上自己的仓库内。

R11：查看完整的版本变迁树

```
git log --graph --oneline --all
```

解释：`git log --graph --oneline --all`命令可以查看完整的版本变迁树，显示所有分支的提交记录及合并情况。

R12：在GitHub上以web页面的方式查看Lab1仓库的当前状态 可以通过浏览器访问GitHub上的Lab1仓库页面，查看仓库中的文件、分支、提交记录等信息。