1.安装 git

Github 服务器管理

1. 在 gitHub 上注册创建用户。 https://github.com/
2. 点击 Start a project 或者 右上角的“+”号。选择 New repository(知识库)
3. 从上至下依次填充 Description、Public、Initialize this repository with a README（可以不勾） 点击：Create repository
4. 弹出页面的 HTTPS SSH 后面的(URL) 即为公司入职时收到的开发地址。 将来向这里提交代码。

本地安装 git 客户端

1. 在 Ubuntu 中执行 git 命令，能看到帮助说明已经安装。
2. 如未安装，执行 apt-get 命令安装 git

2.创建本地代码仓库

1). mkdir 目录 mygit， 进入。 执行：git init。 ls -a 发现多了一个.git 隐藏目录。表成功。

2). vim 创建一个 README.md (md 表 markdown 格式)，使用 git add README.md 命令将 README 添加到本地列表中（成功无提示）

3). git commit README.md -m "首次提交 README.md 文件"，如果是首次提交，应该会出现如下问题。

```
*** Please tell me who you are.
Run
  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"
to set your account's default identity.
Omit --global to set the identity only in this repository.
fatal: unable to auto-detect email address (got 'itcast@itcast.(none)')
```

解决方法：

1. 执行：git config user.name "itcast"
2. 执行：git config user.email "luoyulong@gmail.com"

3. 再次执行上面的提交指令：git commit README.md -m "首次提交 README.md 文件" 可通过。提示如下：

```
[master （根提交） 8bf2edb] 首次提交 README.md 文件
1 file changed, 5 insertions(+)
create mode 100644 README.md
```

4). 指定 github 上对应仓库的 URL：git remote add origin https://github.com/afei-itcastedu/test_redmoo.git

反复执行该命令，会提示：

```
fatal: 远程 origin 已经存在。
```

5). 执行：git status 查看，提示如下：

```
位于分支 master
无文件要提交，干净的工作区
```

6). 执行：git push -u origin master 将本地的内容 推送到 github 上，提示如下：

```
Username for 'https://github.com':                          —— 输入注册的github username
Password for 'https://afei-itcastedu@github.com':           ——输入密码
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 292 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/afei-itcastedu/test_redmoo.git
 * [new branch]      master -> master
分支 master 设置为跟踪来自 origin 的远程分支 master。
```

7). 此时可刷新 浏览器中的 github 会发现页面中多了 README.md 文件。

3.添加代码到 Github 中

1. 将待上传的代码文件拷贝至 git 本机目录中。

2. 执行 git status，将本地与远程做个对比，可看到如下提示：

```
位于分支 master
您的分支与上游分支 'origin/master' 一致。
未跟踪的文件:
  （使用 "git add <file>..." 以包含要提交的内容）
Makefile
include/
main.c
```

```
redmoo.c
提交为空，但是存在尚未跟踪的文件（使用 "git add" 建立跟踪）
```

3. 对将要委托 git 进行管理的文件执行 add 命令： git add main.c 再看 git status。

```
位于分支 master
您的分支与上游分支 'origin/master' 一致。
要提交的变更：
  （使用 "git reset HEAD <file>..." 撤出暂存区）
新文件:       main.c
未跟踪的文件：
  （使用 "git add <file>..." 以包含要提交的内容）
Makefile
include/
redmoo.c
```

4. 可单独提交 main.c 文件： git commit main.c -m "添加 main.c 文件"

```
[master 470caf6] 添加 main.c 文件
 1 file changed, 57 insertions(+)
 create mode 100644 main.c
```

5. 再次查看 git status 。

```
位于分支 master
您的分支领先 'origin/master' 共 1 个提交。
  （使用 "git push" 来发布您的本地提交）
未跟踪的文件：
  （使用 "git add <file>..." 以包含要提交的内容）
Makefile
include/
redmoo.c
提交为空，但是存在尚未跟踪的文件（使用 "git add" 建立跟踪）
```

6. 提交新添加的代码，到远程 github 中：git push -u origin master。 可在浏览器中刷新，查看。

7. 一次提交目录及多个文件到 github 中：

```
1. git add redmoo.c include/
2. git commit redmoo.c include/ -m "添加 redmoo.c 文件和 include 目录"
3. git status
4. git push -u origin master
5. 浏览器中刷新查看。
```

也可以 git commit -a -m "统一提交初始文件" 一次提交多个，但不推荐这么操作。

4.修改代码

1. 在目录中修改代码，如 README.md 中加入一行。

2. 执行 git status 可看到“修改： README.md” 提示。

3. git diff [README.md] 可以查看本地和远程的区别

4.**【非常重要】：**修改代码时，每次在 push 之前，都应该执行一次 git pull 操作。表示将 github 上的代码同步到本地。

无论如何一定要先执行 git pull 一次再 push 操作。以防止，工友已经提交的代码，被你的 push 操作覆盖 !!!

当执行 git pull 时， 提示：Already up-to-date. 说明可无误提交。放心执行 git commit/push 即可。

```
Already up-to-date.
```

5.代码冲突

场景一： 本地代码和 github 冲突。

1. 制造测试环境，手动在 github 上修改一个 README.md 的代码。

2. 同时刻，本地也在 README.md 相同位置修改不同的代码内容。

3. 此时本地 git status 仅仅提示 README.md 修改了。执行 git diff README.md 也看不到远端的内容。

4. 执行 git pull ， 发现没有得到提示：Already up-to-date. 而出现如下内容：

```
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
来自 https://github.com/afei-itcastedu/test_redmoo
   603abe1..846a955  master     -> origin/master
更新 603abe1..846a955
error: Your local changes to the following files would be
overwritten by merge:
    README.md
Please, commit your changes or stash them before you can merge.
Aborting
```

解决冲突步骤：

1. 执行 git stash 。此时再看 README.md，发现之前对 README.md 做的修改不

见了。本地恢复到 README.md 上一个版本。

2. 执行 git pull。将远端 github 上最新文件，拉到本地 README.md 的旧版本上。

3. 执行 git stash pop。 将本地最新修改 与 远程最新修改 同时放置于本地最新 README.md 文件中。 如：

```
<<<<<<< Updated upstream
    这个是 github 修改的代码片段。
=======
    这是本地改修的 README.md,会不会与远程有冲突呢？
>>>>>>> Stashed changes
```

4. 根据实际需求，手动修改文件。然后按照正常提交代码流程提交至远端 github：

```
1). git commit README.md -m "解决冲突后提交 README.md"
2). git pull
3). git push -u origin master
```

场景二： 若 pull 时没问题，push 的一刹那，代码与工友的 push 冲突了。无能为力。极小概率事件，一般为防止此问题，线下提前通知。

6.git 分支

1. 下载 github 上的源码：

1). 可以直接在浏览器中点击 Clone or download --> Download ZIP 保存到本机。

2). 在本地自定义目录内执行：git clone https://github.com/DaveGamble/cJSON.git

2. 分支的概念

分支，一般是基于一份 基础性的泛化源码 创建一个或多个分支，用于各种不同上层业务逻辑的拓展。

3. 使用分支

1). 在 github 上，创建一个分支出来。

2). git branch 在本机查看默认使用的分支

3). git pull 当 github 上有一个新分支的时候，执行 pull 会看到这分支。

```
来自 https://github.com/afei-itcastedu/test_redmoo
 * [新分支]         branch1    -> origin/branch1
Already up-to-date.
```

4). git branch -a 可查看，当前一共有多少分支。

5). git checkout branch1（不建议使用）可将本地分支切换为 branch1。 可用 git branch -a 查看：

```
* branch1
  master
  remotes/origin/branch1
  remotes/origin/master
```

但是，我们强烈不推荐在一个目录中反复切换分支！实在有需求可新建目录，cd 进去，在其中 clone 新分支。

6). git clone -b branch1 https://github.com/afei-itcastedu/test_redmoo.git 可以下载一个分支到当前目录。

7). 修改 README.md，添加 “branch1”一行，再提交到 branch1 分支中，执行流程如：

```
1. git status
2. git pull
3. git commit README.md -m "添加了 branch1"
4. git pull
5. git status
6. git push -u origin branch1
7. 浏览器中查看，branch1 分支中有“branch1”这行，而分支 master 中没有。
```