Git 一分钟上手

流程:取代码 → 每次工作前更新代码到最新版本 → 修改代码 → 提交代码到服务器

取代码及修改全局设置

设置用户名与邮箱

```
1 git config --global user.name "My Name"
2 git config --global user.email "my@email.com"
```

从已有的 git 库中提取代码

```
git clone git@server:app.git myrepo
```

每次更改代码的操作

更新本地代码到最新版本（需要 merge 才能合到本地代码中）

```
git fetch
```

合并更新后的代码到本地

```
git merge
```

更新代码方式的另一种方法(git pull 是 git fetch 和 git merge 命令的一个组合)

```
git pull
```

修改代码后，查看已修改的内容

```
git diff --cached
```

将新增加文件加入到 git 中

```
git add file1 file2 file3
```

从 git 中删除文件

```
1 git rm file1
2 git rm -r dir1
```

提交修改

```
git commit -m 'this is memo'
```

如果想省掉提交之前的 git add 命令，可以直接用

```
git commit -a -m 'this is memo'
```

commit 和 commit -a 的区别, commit -a 相当于：

- 第一步：自动地 add 所有改动的代码，使得所有的开发代码都列于 index file 中
- 第二步：自动地删除那些在 index file 中但不在工作树中的文件
- 第三步：执行 commit 命令来提交

提交所有修改到远程服务器，这样，其它团队成员才能更新到这些修改

```
git push
```

其它常用命令

显示 commit 日志

```
git log
```

不仅显示 commit 日志，而且同时显示每次 commit 的代码改变。

```
git log -p
```

回滚代码：

```
git revert HEAD
```

你也可以 revert 更早的 commit，例如：

```
git revert HEAD^
```

销毁自己的修改

```
git reset --hard
```

查看最新版本和上一个版本的差异(一个^表示向前推进一个版本)

```
git diff HEAD HEAD^
```

将 branchname 分支合并到当前分支中。(如果合并发生冲突,需要自己解决冲突)

```
git merge branchname
```

解决冲突

当 merge 命令自身无法解决冲突的时候，它会将工作树置于一种特殊的状态，并且给用户提供冲突信息，以期用户可以自己解决这些问题。当然在这个时候，未发生冲突的代码已经被 git merge 登记在了 index file 里了。如果你这个时候使用 git diff，显示出来的只是发生冲突的代码信息。

在你解决了冲突之前，发生冲突的文件会一直在 index file 中被标记出来。这个时候，如果你使用 git commit 提交的话，git 会提示：filename.txt needs merge

在发生冲突的时候，如果你使用 git status 命令，那么会显示出发生冲突的具体信息。

在你解决了冲突之后，你可以使用如下步骤来提交：

第一步(如果需要增加文件)：

```
git add file1
```

第二步：

```
git commit
```

git 恢复删除了的文件

git pull 从 git 服务器取出，并且和本地修改 merge， 类似于 SVN up，但是对删除的文件不管用，恢复删除文件用

```
git checkout -f
```