GIT 流程规范

1. master/develop/feature/hotfix 固定分支

master 分支为程序主干目录，开发新需求需从 master 打新分支，严禁直接在 master 上修改代码

develop 为开发分支，开发完成后合并回 master

feature 用于需求开发完成后，已经 merge 到 master，临时出现新需求或者功能变更，需从 master 上新建一个 feature 分支进行开发

hotfix 用于 master 修复分支

2. 新建分支并同步分支

每次开发新功能，都由个人新建一个分支（目前来说我们大家都有自己的分支，分支命名修改为 ”develop(固定分支名作为前缀)” + “_” + “姓名首字母小写”）

新分支通过 pull develop 分支的代码进行同步

3. 提交分支

个人分支修改后，个人测试功能完善后就可以提交 commit 了

提交 commit 时，尽可能给出完整扼要的提交信息，提交信息摘要控制在 50 字以内（模板如下）

```
Present-tense summary under 50 characters

* More information about commit (under 72 characters).
```

```
4.  * More information about commit (under 72 characters).
```

第一行为 50 字以内的摘要，然后空一行，罗列出主要改动、改动原因、需要注意事项（对后续开发或者其他功能的影响），其中摘要与主要改动必填

Commit 格式

```
<type>(<scope>): <subject> <BLANK LINE> <body>
<BLANK LINE>
<footer>
```

type: 必填

➢ feat:新功能

➢ fix:修复 bug

➢ docs:文档变动

➢ style:格式调整，对代码实际运行没有改动，例如添加空行、格式化等

➢ refactor:重构

➢ perf:提升性能的改动

➢ test:添加或修正测试代码

➢ chore:构建过程或辅助工具和库(如文档生成)的更改

subject:必填， 具体的修改描述信息

scope:修改范围，选填。主要是这次修改涉及到的部分，简单概括，例如 login、train-order

```
例如: git commit -m ‘feat(module): <oss> add sts authority management‘
```

git commit --amend 指令追加改动

4. rebase 合并

分支开发完成可能有一堆 commit，不利于后期 code review 更有可能造成分支污染，因此需要用 rebase 命令将多次 commit 命令合并成一次 commit

5. 推送到远程仓库

合并 commit 后通过 push 命令推送到远程分支并@相关人员进行 merge

6. merge

相关人员将最新分支 merge 到 develop 分支，经过测试没问题最终 merge 到 master 进行项目上线