

一、commit 规范

1. 格式规范

`type(功能模块): message`

type

用于说明 commit 的类别，只允许使用下面7个标识。

- `feat`: 新功能 (添加新特性)
- `fix`: 修补bug
- `test`: 增加测试
- `docs`: 文档 (仅仅修改了文档)
- `style`: 格式 (仅仅修改了空格、格式缩进、都好等等，不改变代码逻辑不影响代码运行的变动)
- `refactor`: 重构 (即不是新增功能，也不是修改bug的代码变动)
- `chore`: 构建过程或辅助工具的变动或者增加依赖库、工具等Commit messages格式要求

举个例子，新增了用户登录功能，那么提交信息应该这样写：

```
feat(user): add the user login
```

其他type提交信息举例：

```
E:\Project\open>git commit -m "ffffx"
git commit -m "feat(新烟入库): 增加垛位分配管理功能"
git commit -m "fix(新烟分流): 修复成品分流列表查询检索条件无效bug"
git commit -m "test(用户管理): 增加用户登录接口junit测试代码"
git commit -m "docs(ReadMe): 增加readme.md 说明"
git commit -m "style(系统管理): 格式化代码, 去掉多余的注释代码"
git commit -m "refactor(菜单管理): 重构菜单管理代码, 接口响应报文统一格式输出"
git commit -m "chore(框架): 修改打包方式, 增加docker打包依赖工具"
```

2. message规范

`type(功能模块): message` 中的message 长度必须大于5个字符并小于100个字符

二、gitlab服务器端校验

push 代码的时候会进行服务端验证，可能会响应比较慢。

1. push校验

当我们提交不规范的commit信息时就会提醒用户，并终止此提交。

```
E:\Project\open>git commit -m "ffffx"
#####
##
## push message style check failed!
##
## type must be one of [feat,fix,docs,style,refactor,test,chore]##
##
## Example:
## feat(user): add the user login.
##
##
#####
```

如果提交的日志不规范则会push校验失败，请自己修改本地提交日志后 重新提交。

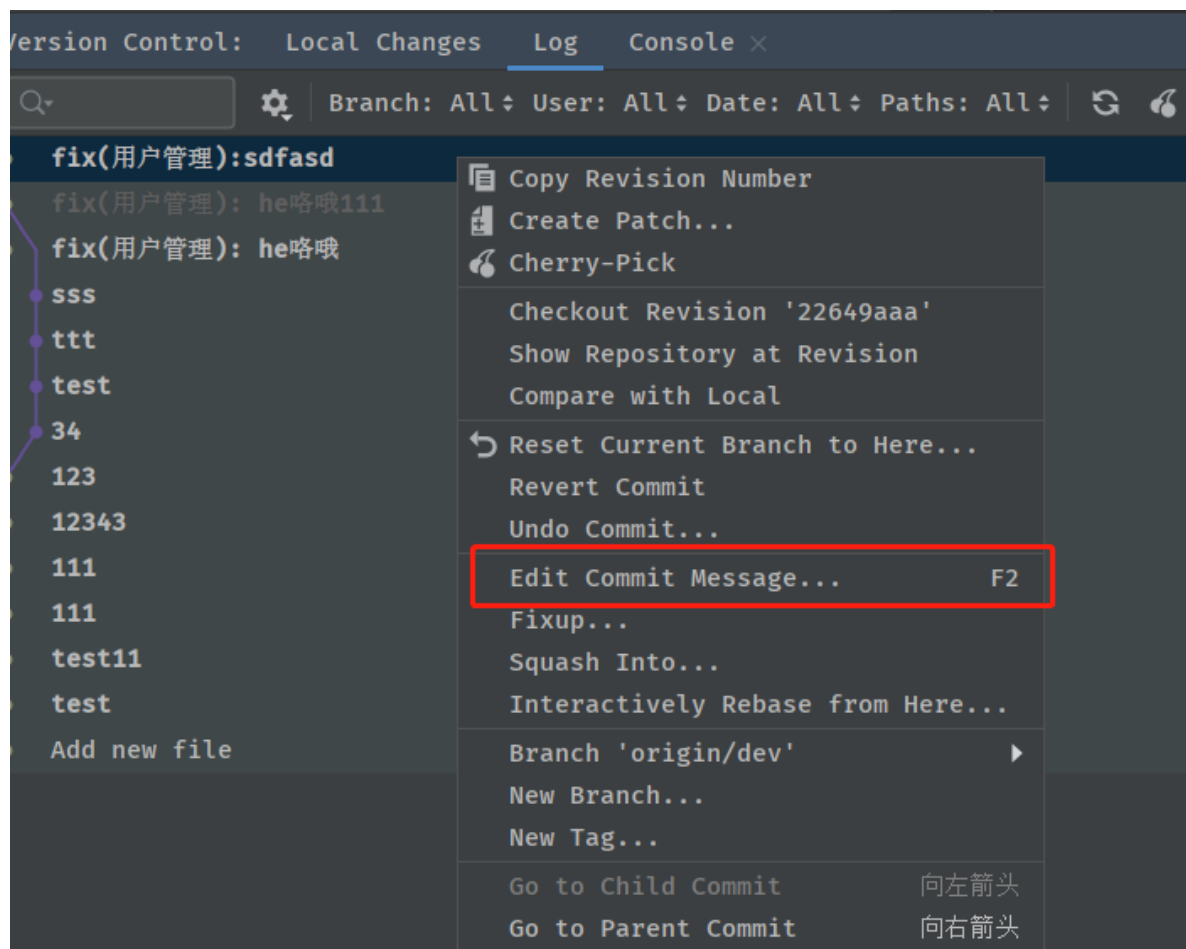
2. 本地message信息修改

本地message修改有两种方式

1. 命令行修改提交信息，执行下面命令后使用vi编辑后保存。

```
git commit --amend
```

2. idea 工具修改提交信息：右键提交的message log，选择Edit Commit Message，修改后再提交日志



三、注意事项

1. 提交代码最好单功能单模块提交，尽量不要组合模块提交，不便于日志信息准确描述。
2. (功能模块)需要是当前项目已存在的模块。

3. message信息尽量简短并描述清楚，不要出现类似“解决bug、修复已知bug、新增功能”等的提交信息。
4. 如果push失败，请仔细阅读push返回回来的错误信息并更正。
5. 如果commit message的格式正确，但是依旧push失败，请联系我

四、其他

1. 后续会增加更多日志提交的检查功能，此文档持续更新。
2. 另，还会将会增加代码检查功能，请大家现在按照阿里巴巴java开发规范书写代码，便于养成习惯，减少后面大家提交代码错误。