首页概览

描述

前端开发手册

manual

前端手册

如需按顺序生成pdf，文件名需加上序号。

分支管理

> 描述
>
> 分支管理及 `git` 工作流

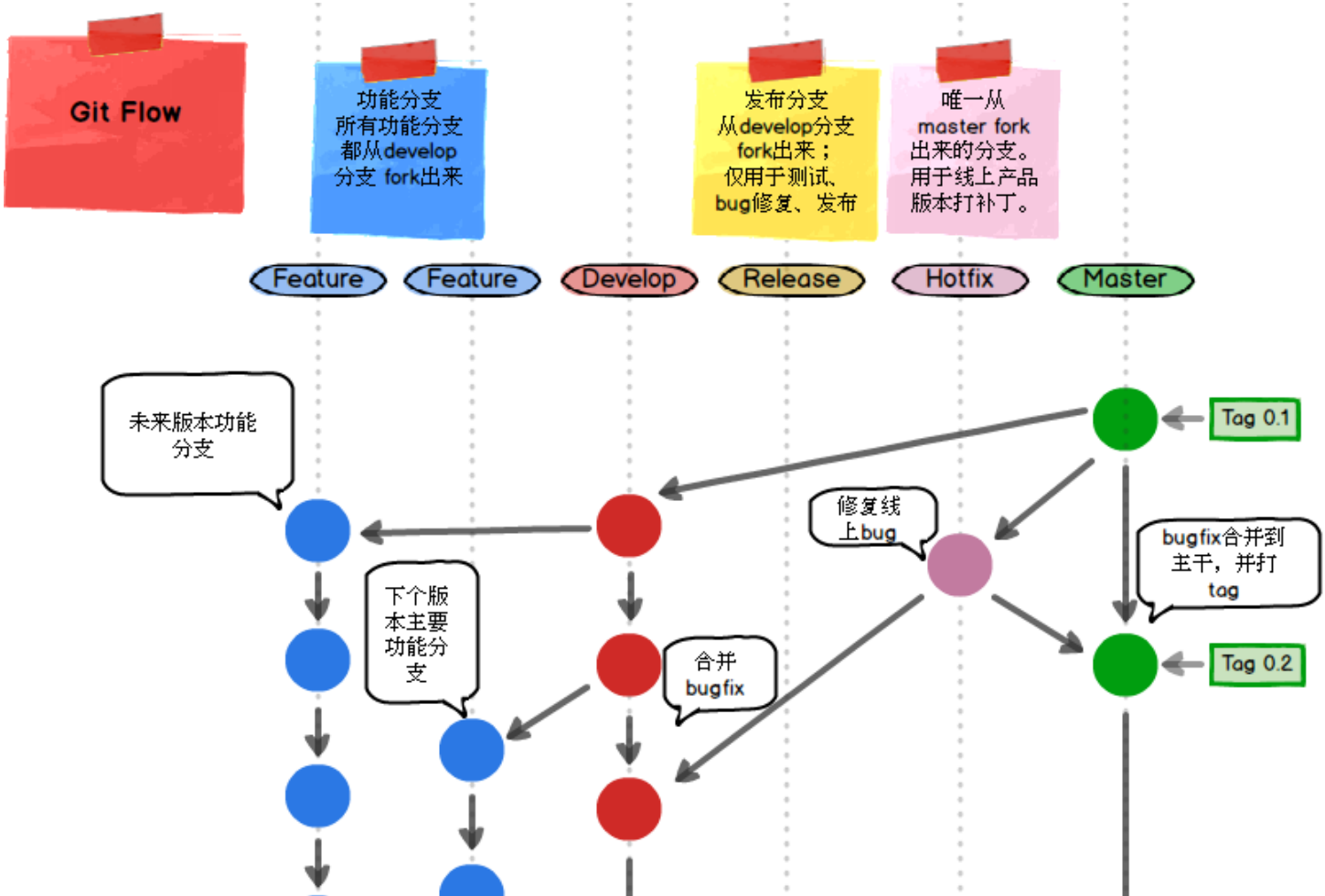
主要分支说明

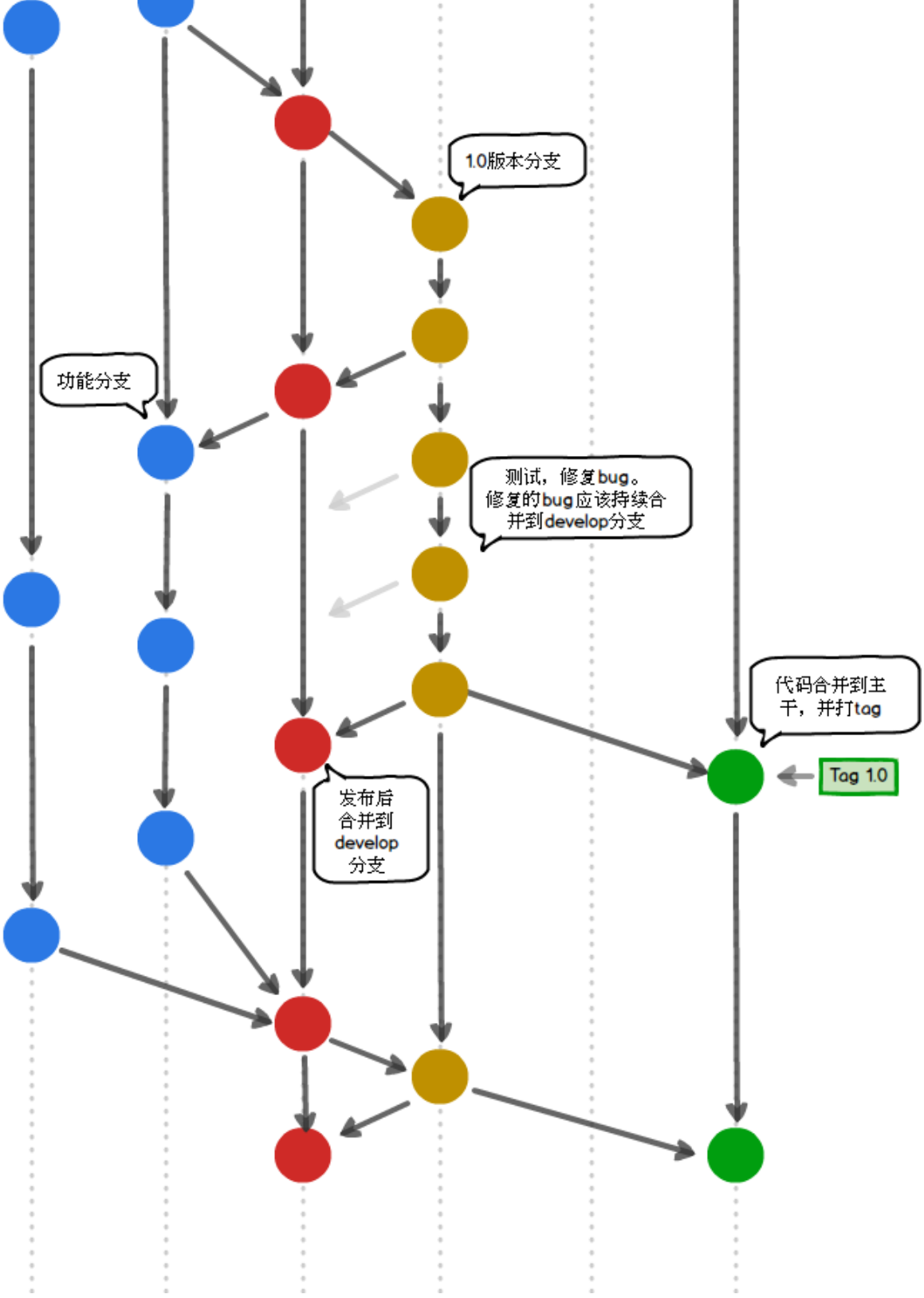
- `master` ： `protected` 分支
- `develop` ： 开发分支，开发人员将 `feature` 分支合并至此
- `feature` ： 功能分支，开发人员新拉取 `feature` 分支进行新功能模块的开发，开发完成测试无误后合并至 `develop` 分支
- `release` ： 用于版本发布
- `hotfix` ： 热修复分支，从 `master` 拉取

feature 命名: feature-[账号名]-[功能模块]

如场景模块的功能开发， `feature-kezai-scenes`

Git Flow




1.0版本分支
功能分支
测试，修复bug。
修复的bug应该持续合
并到develop分支
代码合并到主
干，并打tag
Tag 1.0
发布后
合并到
develop
分支

提交规范

> 描述
>
> 统一定义管理代码提交规范，方便 持续集成发布 `tag` 、 `release` 以及生成 `chagnelog` 文档，使用 `Angular commit` 规范标准，细节查看 `https://github.com/angular/angular/blob/master/CONTRIBUTING.md#-commit-message-guidelines`

基本格式

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

`Type` 类型：

- `feat` ： 新功能（ `feature` ）
- `fix` ： 修补 `bug`
- `docs` ： 修改文档（ `documentation` ）
- `style` ： 修改格式，如标签、空格、格式化、分号等（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）
- `test` ： 增加测试
- `chore` ： 构建过程或辅助工具的变动

通常 `feat` 和 `fix` 会被放入 `changelog` 中，其他(`docs` 、 `chore` 、 `style` 、 `refactor` 、 `test`) 通常不会放入 `changelog` 中。

提交范例

> 尽可能详细描述提交的内容

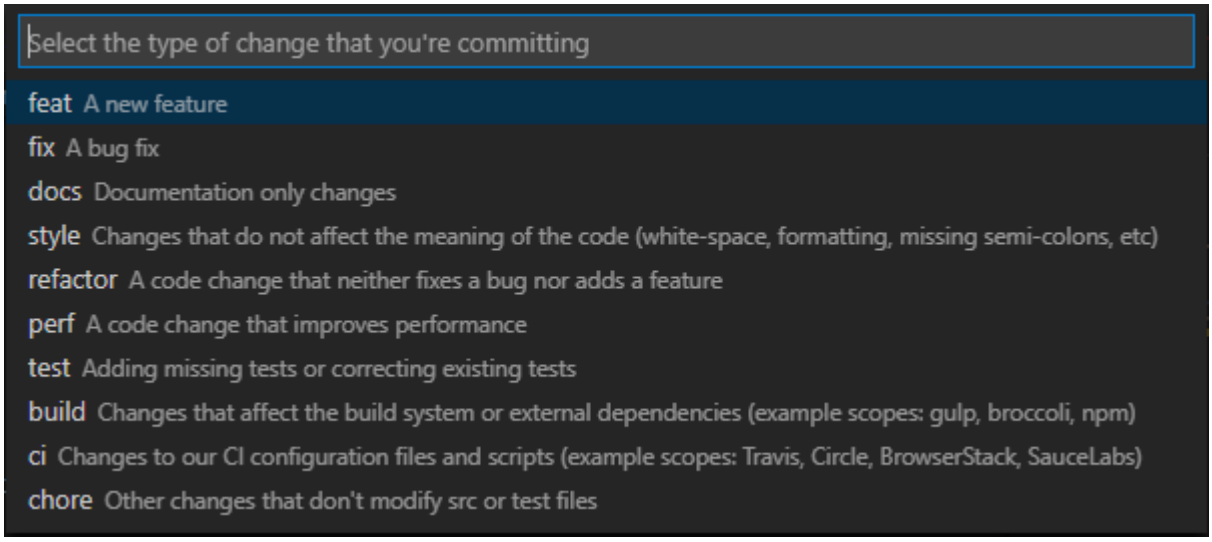
```
feat: create a new feature

fix: 修复登录页在IE10浏览器布局错乱的问题

fix: add vuepress-plugin-export-pdf in config.js
```

```
docs: generate manual docs
```

其他

可以安装 `vscode-commitizen` 插件，使用 `ctrl+shift+p` 或 `command+shift+p` 使用 `conventional commit` 提交代码。

`optional` 的选项可以忽略输入。



样式管理

> 描述
>
> 使用 `BEM` 方法组织定义样式，以便结构化、唯一，避免样式冲突互相污染等，详情查看bem

BEM

> `block-name__element-name--modifier-name` ：模块名 + 元素名 + 修饰器名

```css
.menu {}
.menu--fixed{}
.menu_item{}
.menu_item--active{}
.menu_item--dsiabled{}
```

Blocks

独立的实体模块， 如 `header` ， `container` ， `menu` ， `checkbox` ， `input` ， `footer` 等

Elements

模块的一部分，与模块紧密关联

如 `header title` ， `menu item` ， `list item` ， `checkbox caption` 等

Modifiers

模块或元素的修饰标志，用来表示外观或行为。

如 `disabled` , `highlighted` , `checked` , `fixed` , `color yellow` , `active` , `size big` 等

小程序框架

描述

小程序框架使用说明

管理平台

描述

管理平台基础框架使用说明

SourceTree基本使用教程

0. 安装

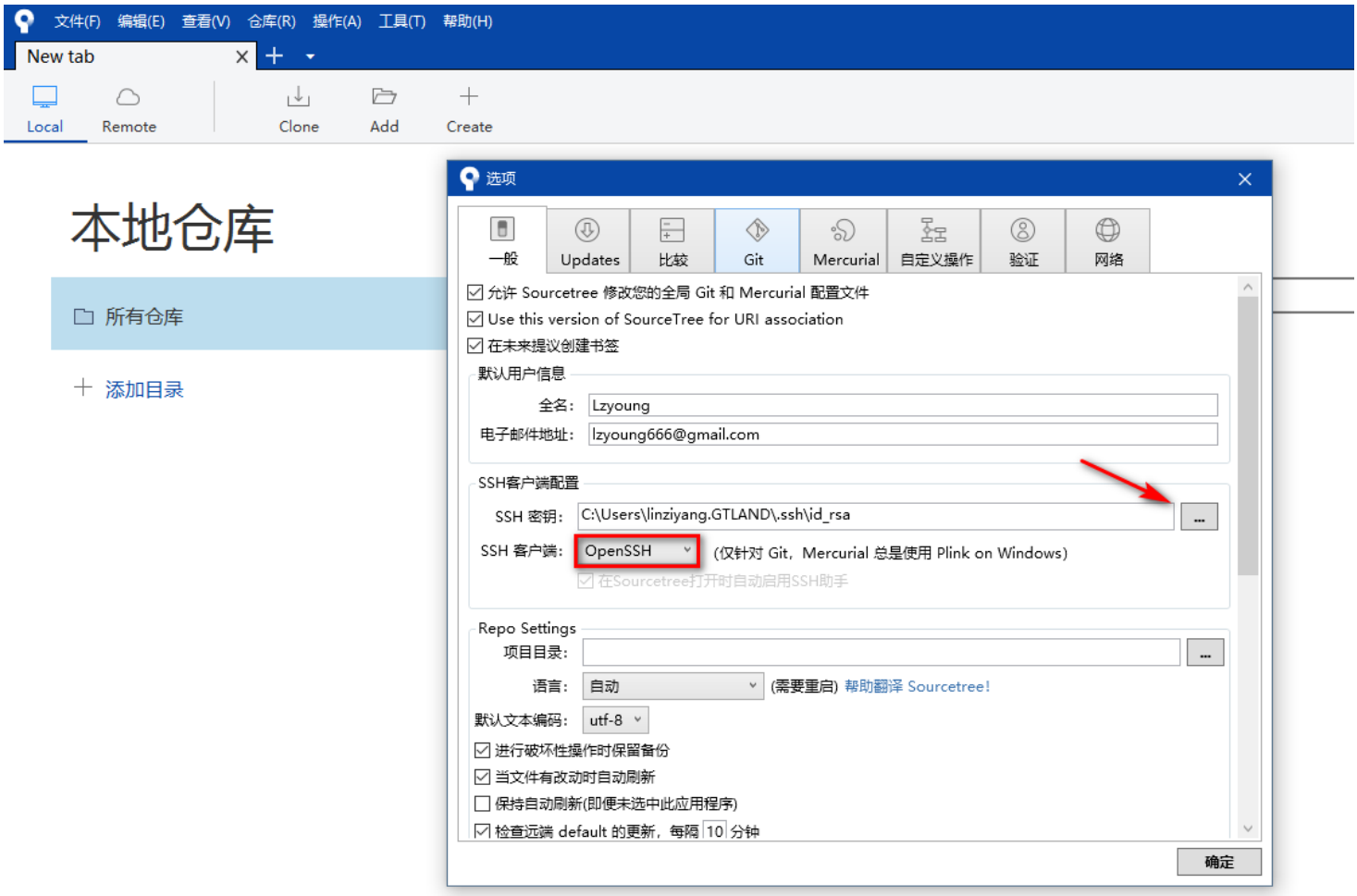
> 下载 SourceTreeSetup 安装包文件

安装步骤：

- 注册账号

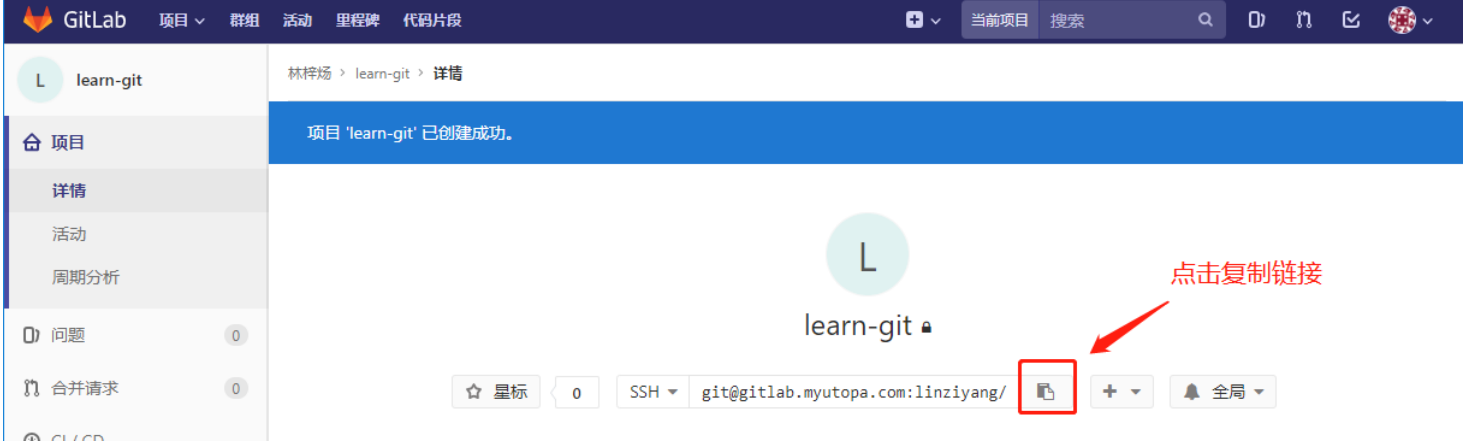
1. 配置 ssh 密钥信息

在菜单栏中【工具】—【选项】中配置已生成的 SSH 密钥（选择 SSH 客户端为 OpenSSH ）



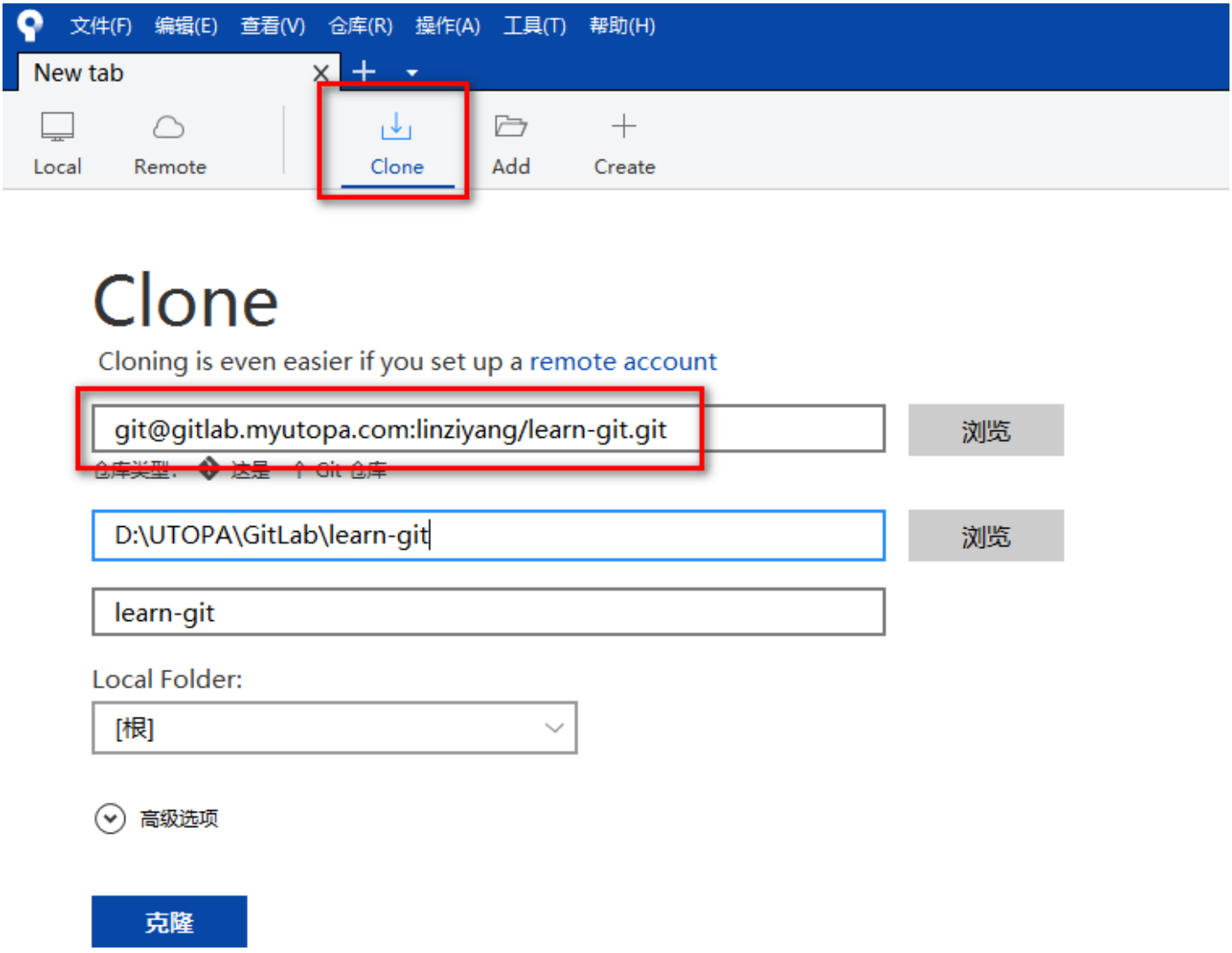
2. Clone 远程仓库至本地

2.1 到 GitLab 中选取需要 clone 的项目，复制链接地址

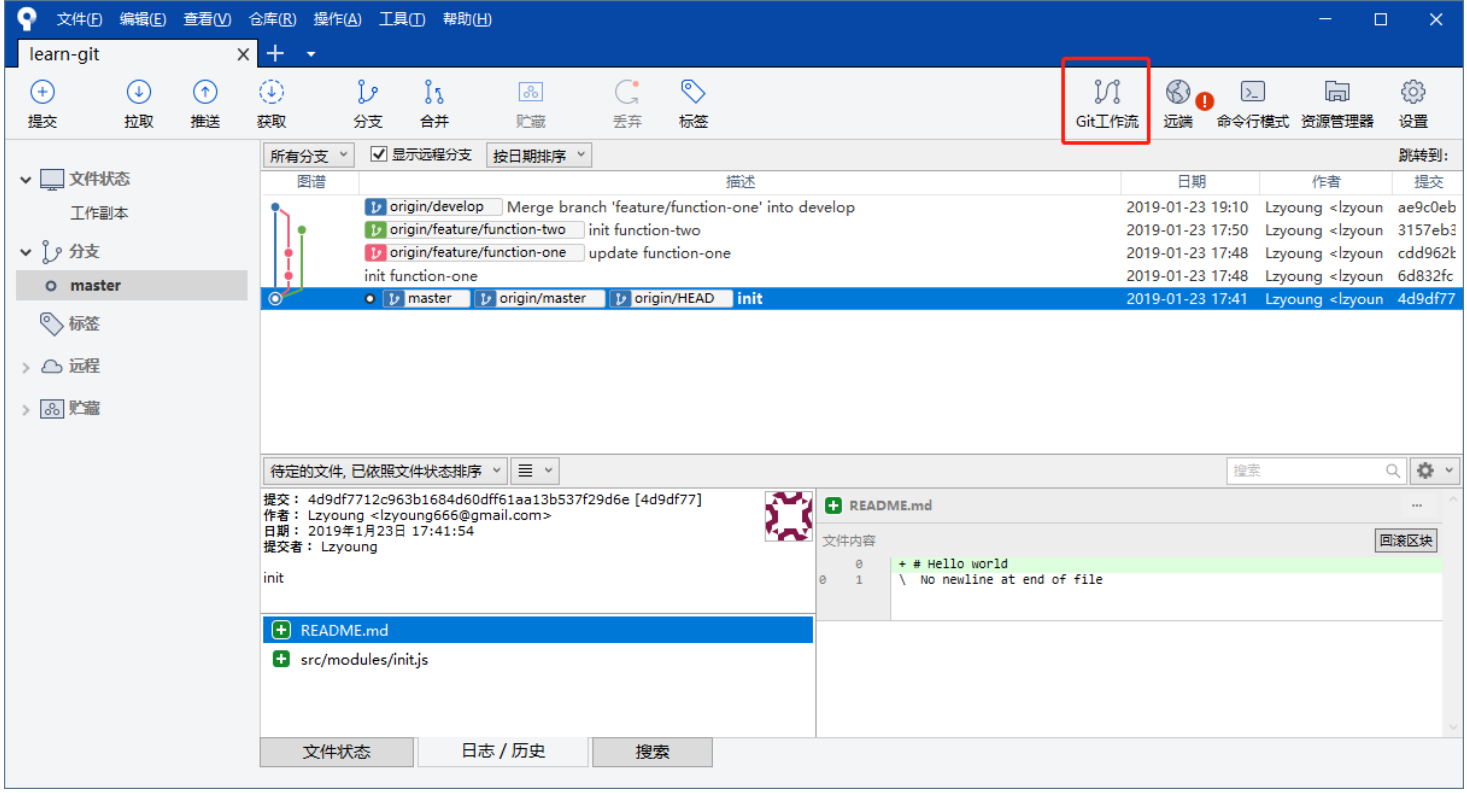


2.2 在 SourceTree 中点击【 Clone 】，填写相应信息，依次为：

- 远端仓库地址
- 本地保存仓库地址
- 本地仓库名称



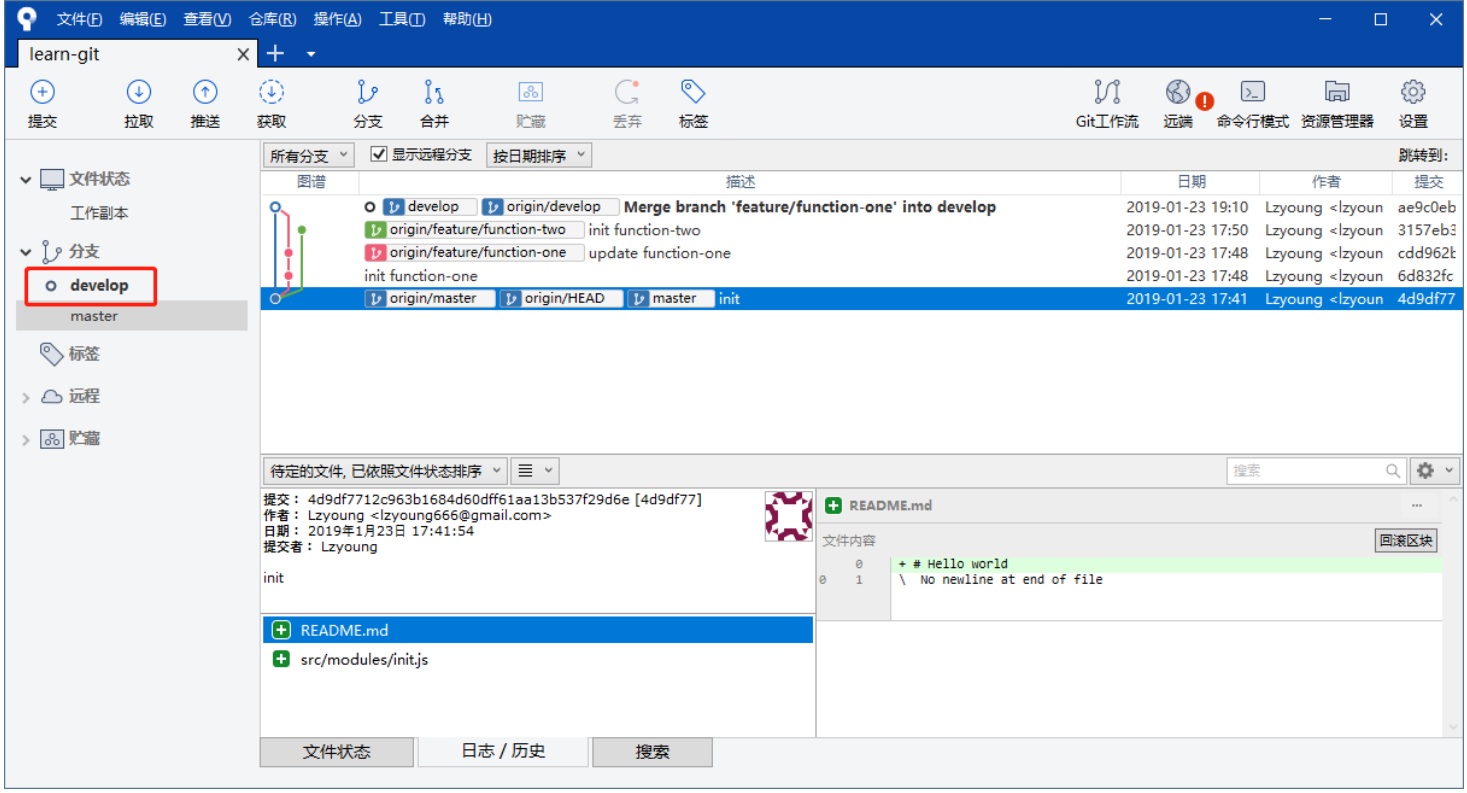
2.3 填写完成后，点击【克隆】按钮，进行clone，完成后如下图：



3. 工作流

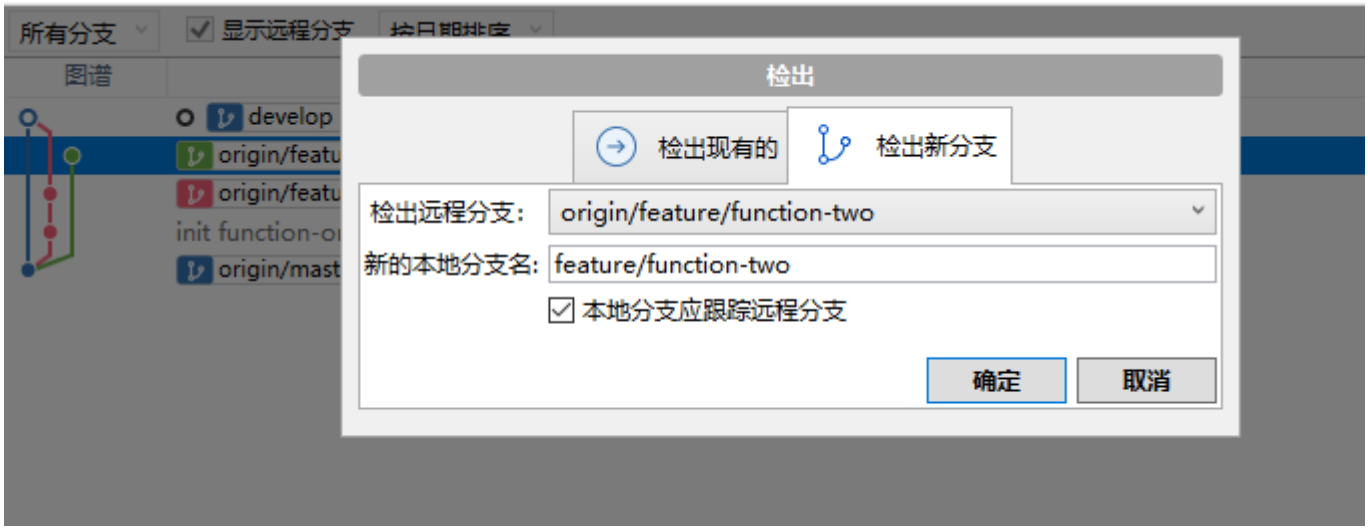
3.1 点击【 Git 工作流】，弹出框中点击【确定】（默认设置）

3.2 在左侧分支树中新增了 develop 分支且项目被默认切换到 develop 分支下

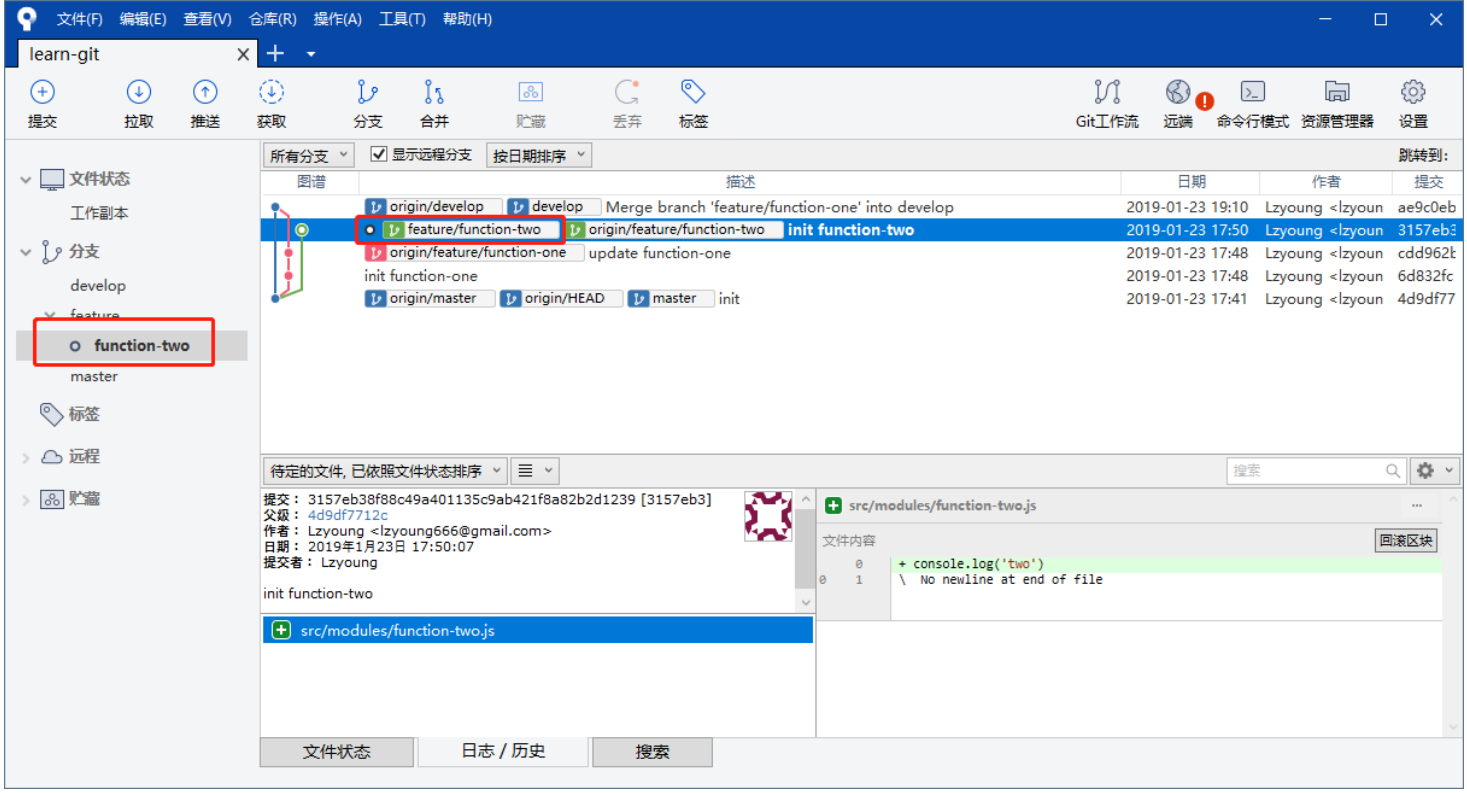


4. 新建 feature 分支

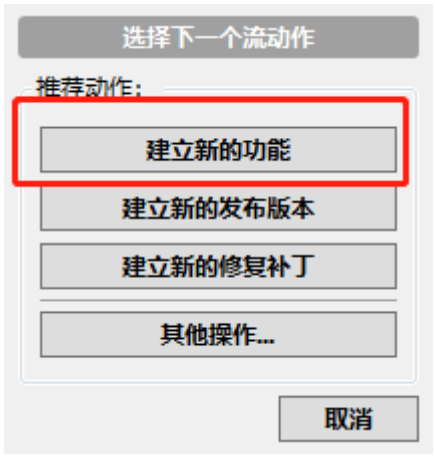
4.1 若分支中有需要继续开发的功能，双击该 feature 检出该功能 feature 分支



4.2 在左侧分支树中新增了 feature 分支且项目被默认切换到继续开发的 feature 分支下



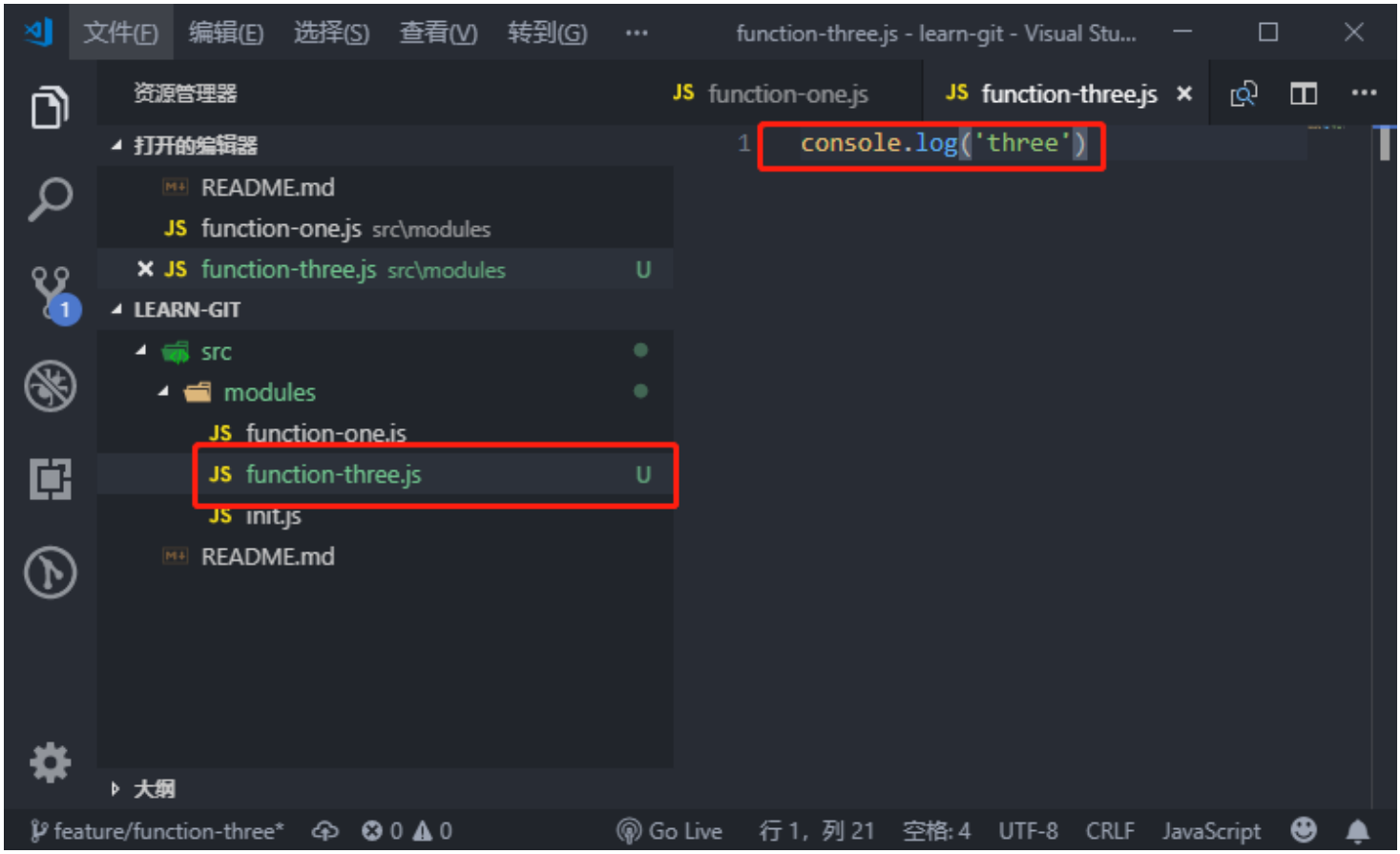
4.3 若要新建功能分支，需要双击左侧分支树中的 `develop` 切换至该分支，点击【 `Git` 工作流】



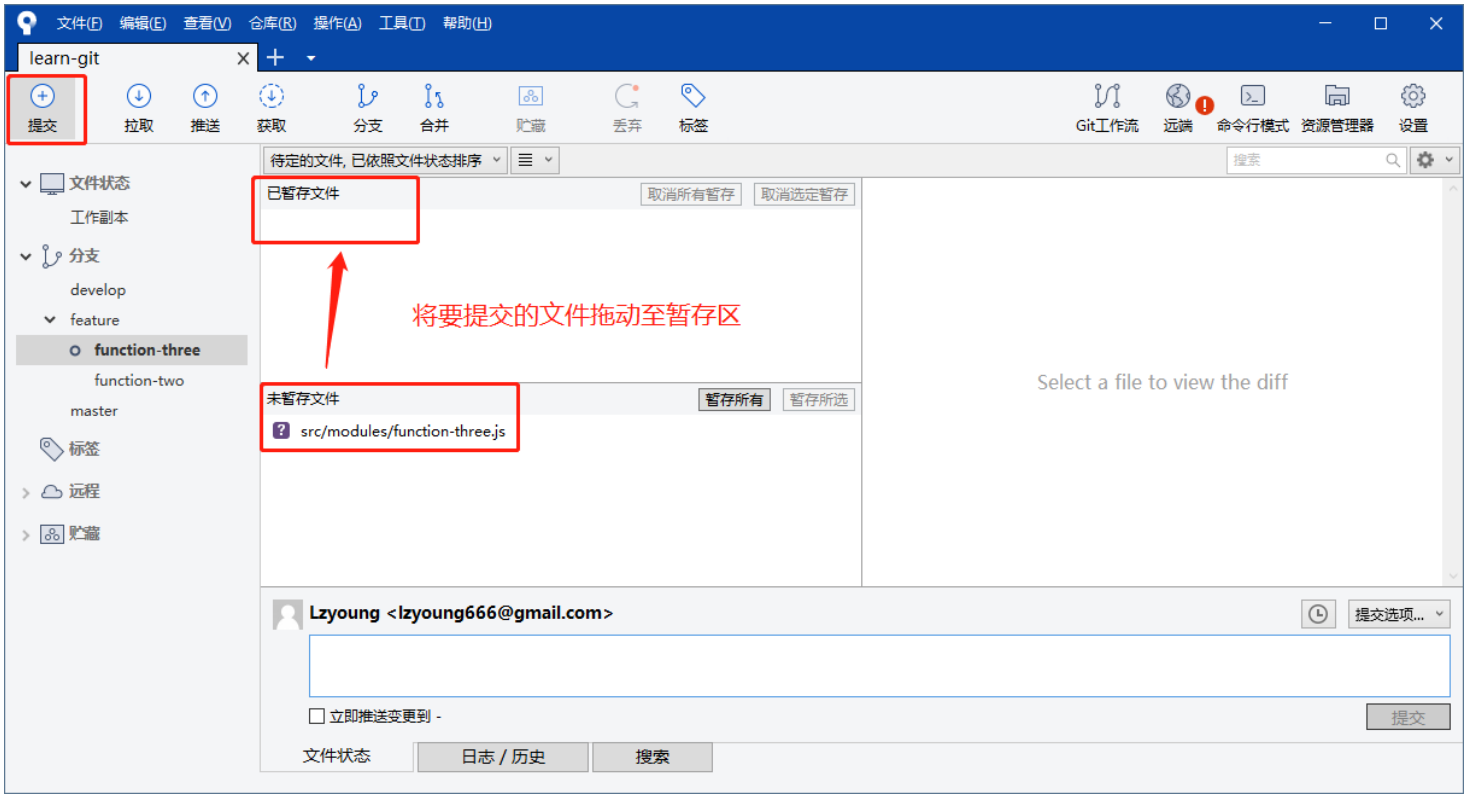
4.4 填写新增功能分支名称，点击【确定】新建功能分支

5. `git add`

5.1 SourceTree 会自动将分支切换到新建的分支。在当前分支中新增内容

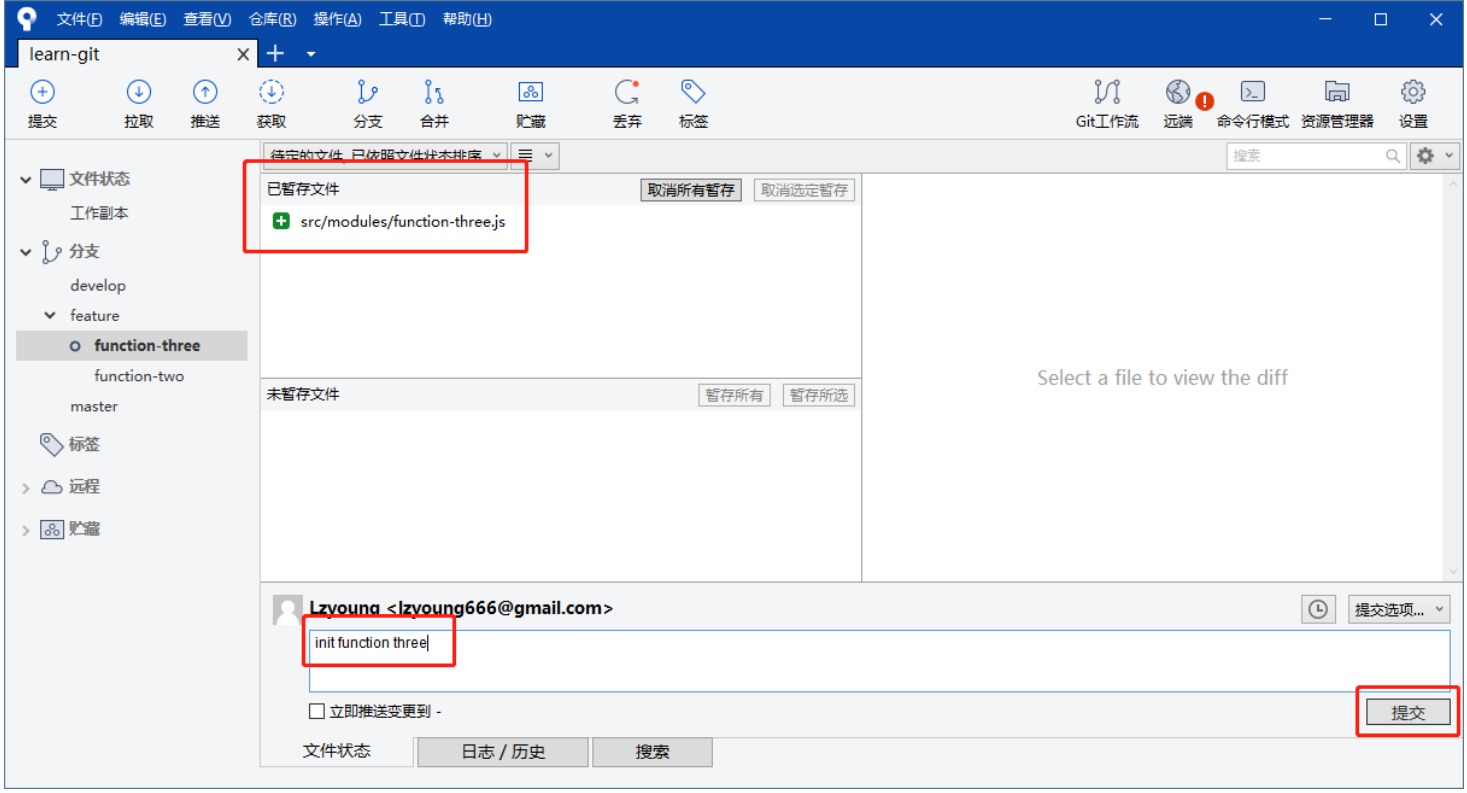


5.2 点击【提交】进入提交页面，将要提交的文件拖动至暂存区（或右键需要提交的文件点击【添加】）进行 add 操作



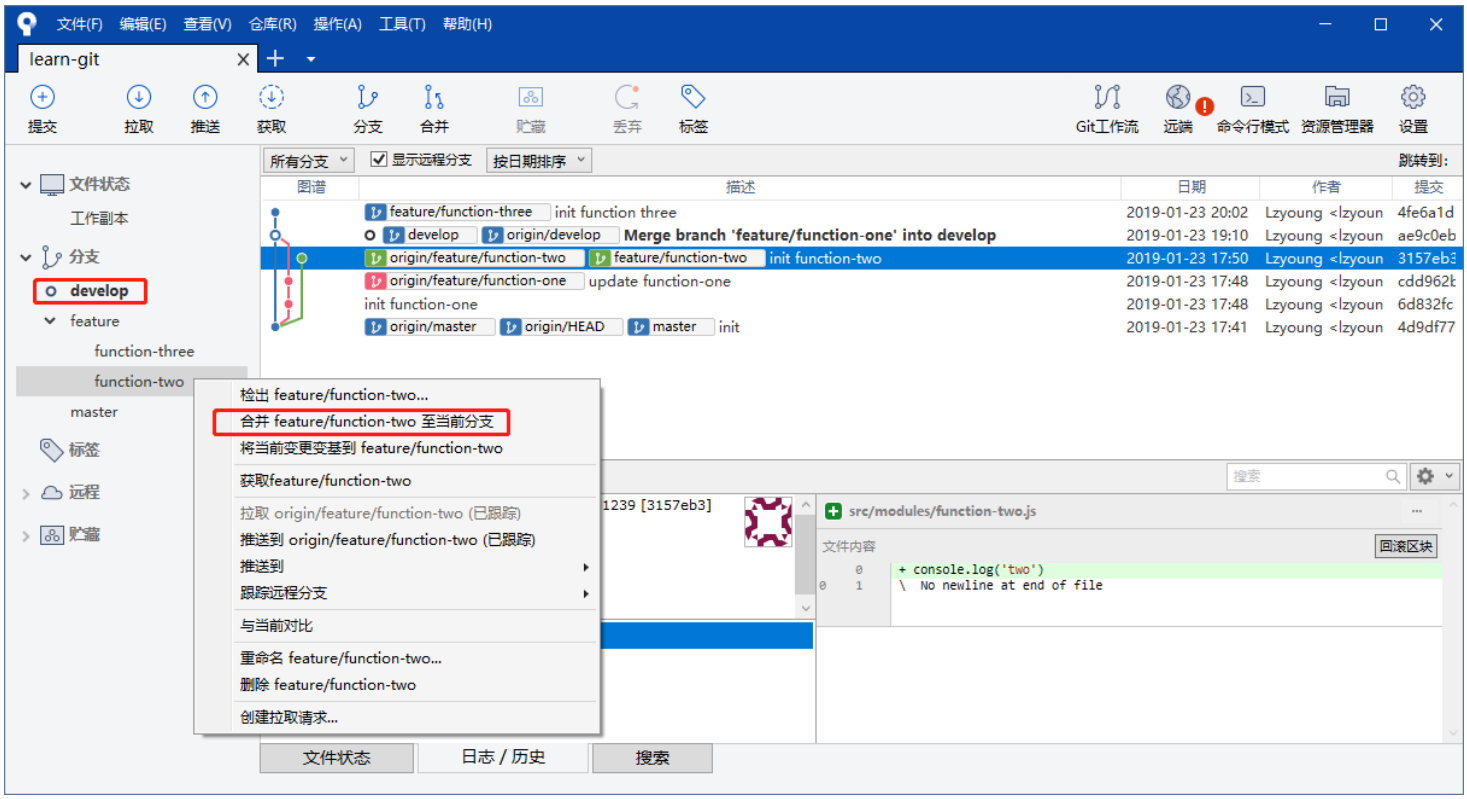
6. git commit

确认需提交的文件后，填写提交信息，点击【提交】进行 commit 操作



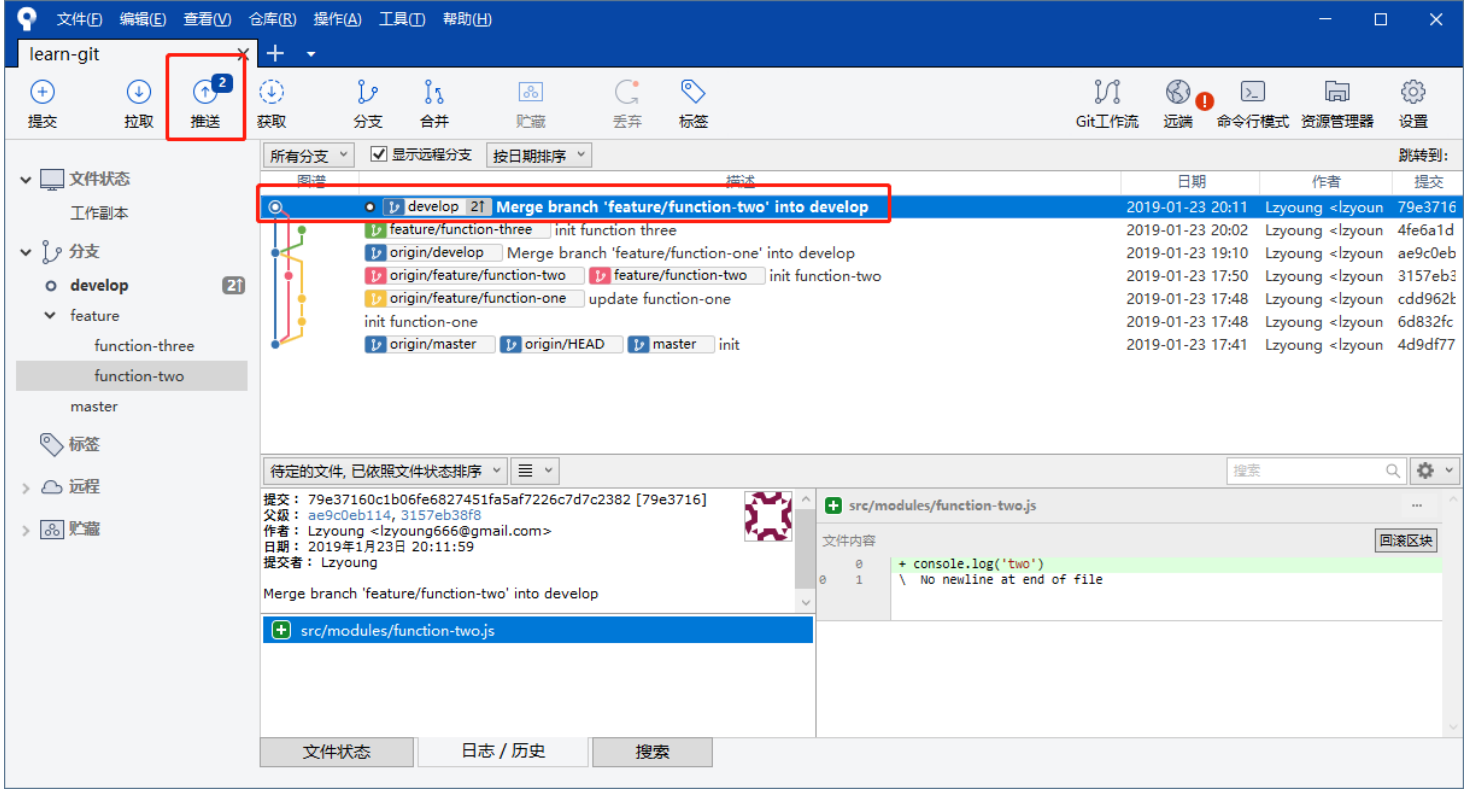
7. git merge

合并已完成功能分支至 develop 分支，需先切换至 develop 分支，右键需要合并至 develop 分支的 feature 分支，选择【合并...至当前分支】

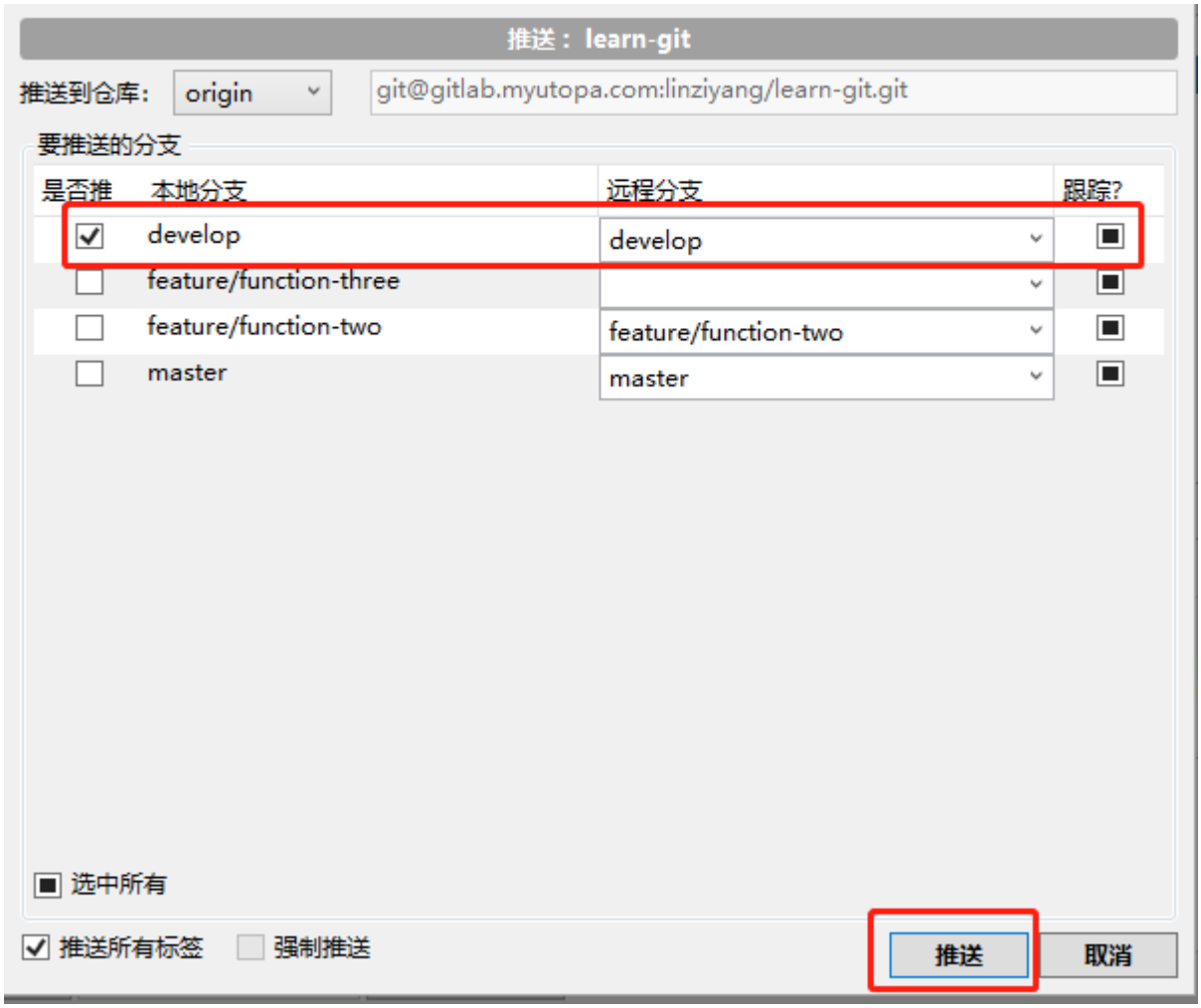


8. git push

8.1 将 develop 分支推送至远程仓库，点击【推送】进行 push 操作

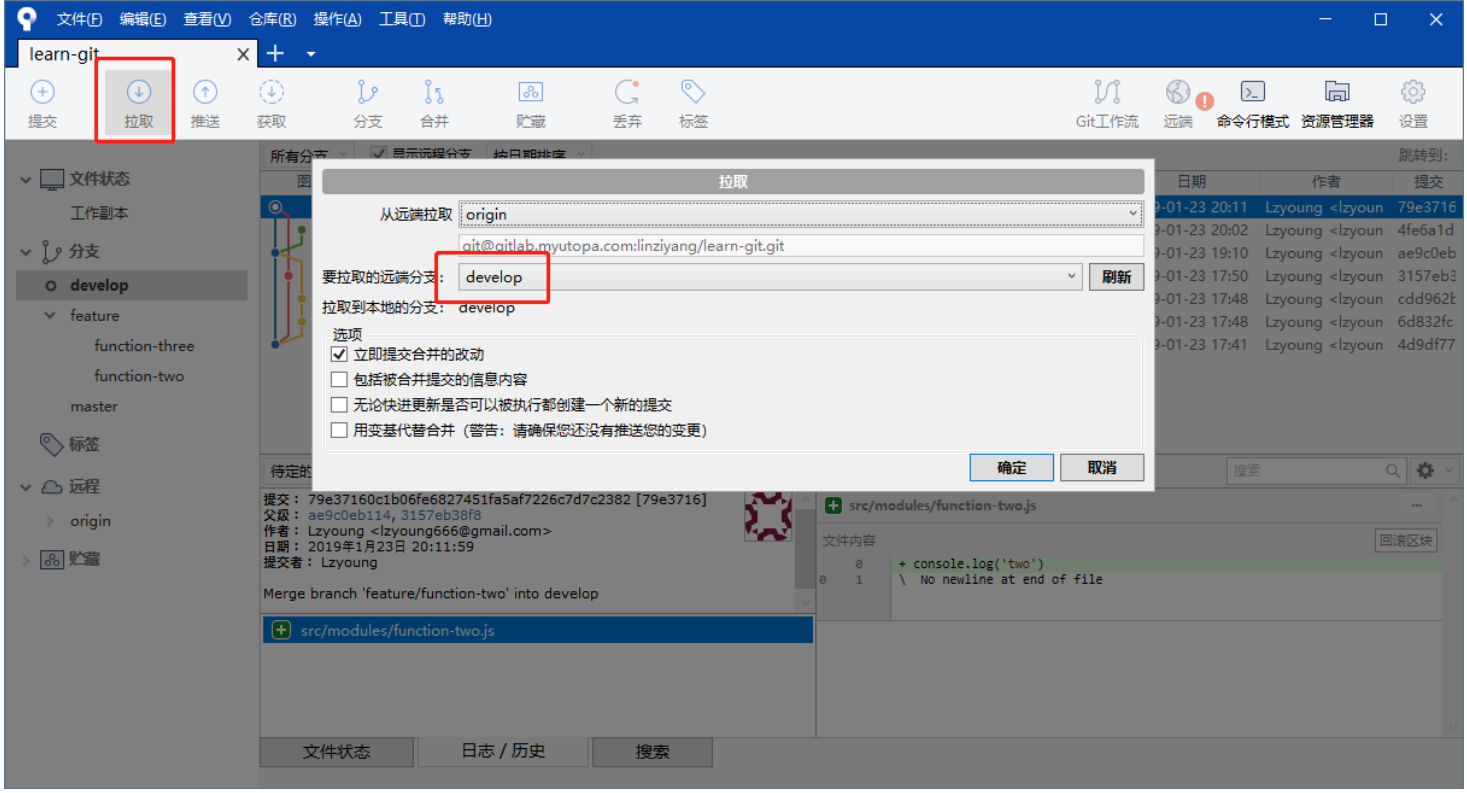


8.2 选择要推送至远程仓库的分支，点击【推送】完成 `push` 操作



9. `git pull`

拉取远程仓库中的分支（每次提交都需要确保已经拉取远程 `develop` 代码）


文件(F) 编辑(E) 查看(V) 仓库(R) 操作(A) 工具(T) 帮助(H)
learn-git
提交
拉取
推送
获取
分支
合并
贮藏
丢弃
标签
Git工作流
远端
命令行模式
资源管理器
设置
文件状态
工作副本
分支
develop
feature
function-three
function-two
master
标签
远程
origin
贮藏
拉取
从远端拉取 origin
git@gitlab.myutopa.com:linziyang/learn-git.git
要拉取的远端分支: develop
刷新
拉取到本地的分支: develop
选项
立即提交合并的改动
包括被合并提交的信息内容
无论快进更新是否可以被执行都创建一个新的提交
用变基代替合并（警告：请确保您还没有推送您的变更）
确定
取消
提交： 79e37160c1b06fe6827451fa5af7226c7d7c2382 [79e3716]
父级： ae9c0eb114, 3157eb38f8
作者： Lzyoung <lzyoung666@gmail.com>
日期： 2019年1月23日 20:11:59
提交者： Lzyoung
Merge branch 'feature/function-two' into develop
src/modules/function-two.js
文件内容
+ console.log('two')
\ No newline at end of file
文件状态
日志 / 历史
搜索

GitHub 和 GitLab 共同使用

说明

> 公司 GitLab 私服与个人 GitHub 服务，因使用邮箱不同，产生的 SSH key 不同，从而导致冲突。

解决方案

方案一

Gitlab 与 Github 邮箱地址保持一致！

方案二

基于 config 文件

1. 配置 Git 用户名、邮箱

> window 系统， ssh 的路径为： C:\Users\Administrator\.ssh

用户 onbright 的配置如下：

```sh
# 全局配置，Github仓库中默认使用此配置
git config --global user.name 'onbright' && git config --global user.email 'onbright@gmail.com

# *团队项目配置，每次新创建一个项目，需要执行下
git config --local user.name 'onbright' && git config --local user.email 'onbright@on-bright.c
```

2. 生成 ssh key 上传到 Github/Gitlab

ssh key 默认生成后保存在 ~/.ssh/ 目录下，默认为 id_rsa 和 id_rsa.pub 两个文件，由于我们需要分开配置，所以这么做：

```sh
# 生成公钥、密钥的同时指定文件名，Gitlab使用
ssh-keygen -t rsa -f ~/.ssh/id_rsa.gitlab -C "onbright@on-bright.cn"

# 生成默认，Github使用
ssh-keygen -t rsa -C "onbright@gmail.com"
```

命令执行完成后，这时 ~/.ssh 目录下会多出 id_rsa.gitlab 和 id_rsa.gitlab.pub 两个文件，id_rsa.gitlab.pub 里保存的就是我们要使用的 key ，这个 key 就是用来上传到 Gitlab 上的。

3. 配置 config 文件

在 ~/.ssh 目录下，如果不存在，则新建 touch ~/.ssh/config 文件**(window 系统在 ssh 目录下右键创建 config 文件)** ，文件内容添加如下：

```sh
Host *.myonbright.com
     IdentityFile ~/.ssh/id_rsa.gitlab
     User onbright
```

配置完成后，符合 *.onbright.com 后缀的 Git 仓库，均采取 ~/.ssh/id_rsa.gitlab 密钥进行验证，其它的采取默认的。

4. 上传 public key 到 Github/Gitlab

以 Github 为例，过程如下：

1. 登录 github
2. 点击右上方的 Accounting settings 图标
3. 选择 SSH key
4. 点击 Add SSH key

在出现的界面中填写 SSH key 的名称，填一个你自己喜欢的名称即可，然后将上面拷贝的 ~/.ssh/id_rsa.pub 文件内容粘帖到 key 一栏，在点击“ add key ”按钮就可以了。

添加过程 github 会提示你输入一次你的 github 密码，确认后即添加完毕。 上传 Gitlab 的过程一样，请自己操作。

5. 验证

```sh
➜  ~  ssh -T git@github.com
Hi onbright! You've successfully authenticated, but GitHub does not provide shell access.
➜  ~  ssh -T git@gitlab.myonbright.com
Welcome to GitLab, onbright!
```

git基本命令

> 描述
>
> git 基本命令行操作

微信分享

> 描述
>
> 微信分享示例

```js
import wx from 'weixin-js-sdk'
import { getWechatConfigFromServer } from 'serverApi'

// 初始化
const WechatInit = async () => {
  const config = await getWechatConfigFromServer()
  return new Promise((resolve, reject) => {
    wx.config({
      debug: false,
      appId: config.appId, // 必填，公众号的唯一标识
      timestamp: config.timestamp, // 必填，生成签名的时间戳
      nonceStr: config.nonceStr, // 必填，生成签名的随机串
      signature: config.signature, // 必填，签名，见附录1
      jsApiList: [ // 必填，需要使用的JS接口列表，所有JS接口列表见附录2
        'onMenuShareAppMessage', // 获取“分享给朋友”按钮点击状态及自定义分享内容接口
        'onMenuShareTimeline', // 获取“分享到朋友圈”按钮点击状态及自定义分享内容接口
        'onMenuShareQQ'
      ]
    })
    wx.ready(res => resolve(res))
    wx.error(res => reject(res))
  })
}

// 微信分享到朋友圈
const ShareTimeLine = (option) => {
  return new Promise((resolve, reject) => {
    wx.onMeuShareTimeline({
      title: opstion.title, // 分享标题
      link: opstion.link, // 分享链接
      imgUrl: opstion.imgUrl, // 分享图标
      success: resolve,
      cancel: reject
    })
  })
}

// 微信分享到朋友
```

```
const ShareAppMessage = (option) => {
  return new Promise((resolve, reject) => {
    wx.onMenuShareAppMessage({
      title: option.title, // 分享标题
      desc: option.desc, // 分享描述
      link: option.link, // 分享链接，该链接域名或路径必须与当前页面对应的公众号JS安全域名一致
      imgUrl: option.imgUrl, // 分享图标
      type: option.type, // 分享类型,music、video或link，不填默认为link
      dataUrl: option.dataUrl, // 如果type是music或video，则要提供数据链接，默认为空
      success: resolve,
      cancel: reject
    })
  })
}

// 微信分享到朋友
const ShareQQMessage = (option) => {
  return new Promise((resolve, reject) => {
    wx.onMenuShareQQ({
      title: option.title, // 分享标题
      desc: option.desc, // 分享描述
      link: option.link, // 分享链接，该链接域名或路径必须与当前页面对应的公众号JS安全域名一致
      imgUrl: option.imgUrl, // 分享图标
      success: resolve,
      cancel: reject
    })
  })
}
```

示例使用

```
const shareConfig = {
  link: 'https://gz.on-bright.com/wx/share/fashionWeek?isApp=0',
  title: 'FASHION PARTY2018•LIBER时尚趴SEE NOW BUY NOW即秀即买',
  desc: 'LIBER时尚周开启即秀即买模式，并结合最新科技展示，同时通过线上直播及各地LIBER时尚周分会场，
  imgUrl: 'https://public.on-bright.com/share_logo_link.png',
}

WechatInit().then(() => {
  ShareAppMessage({
    title: shareConfig.title,
    desc: shareConfig.desc,
    link: shareConfig.link,
    imgUrl: shareConfig.imgUrl,
    type: 'link',
    dataUrl: ''
  })
  ShareTimeLine({
```

```
    title: shareConfig.title,
    link: shareConfig.link,
    imgUrl: shareConfig.imgUrl
  })
  ShareQQMessage({
    title: shareConfig.title,
    link: shareConfig.link,
    imgUrl: shareConfig.imgUrl
  })
})
```