# 基于CODING的aerocar开发流程

编写：

校验：

审核：

修订页

|  |  |  |
| --- | --- | --- |
| 版本 | 内容 | 修订人 |
| V1.0 | 针对aerocar项目的开发流程 | 黄潇剑 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 1.环境搭建

### 1.1安装ubuntu20.04

### 1.2注册CODING，进入团队

#### 1.2.1注册CODING

参考链接：[注册与邀请 - 什么是 DevOps? DevOps 介绍 ｜ CODING DevOps](https://coding.net/help/docs/start/register-invite.html)

#### 1.2.2由团队负责人邀请进入团队

### 1.3配置aerocar

#### 1.3.1安装GIT

在根目录下打开终端，输入指令：sudo apt install git

#### 1.3.2克隆aerocar代码

方法一：输入账号和密码

在根目录下打开终端，输入指令：

git clone https://e.coding.net/ifccq/aerocar/AeroCar.git --recursive（需要你的CODING账号和密码）

方法二：SSH公钥

参考链接：https://coding.net/help/docs/repo/ssh/config.html

#### 1.3.3更新仓库子模块

在根目录下打开终端，输入指令：

cd AeroCar

git submodule update --init --recursive

cd src/drivers/uavcan/libuavcan

git remote add IFC https://e.coding.net/ifccq/aerocar/libuavcan.git（需要你的CODING账号和密码）

git fetch IFC

cd mavlink/include/mavlink/v2

git remote add IFC https://e.coding.net/ifccq/aerocar/mavlink\_c\_library\_v2.git（需要你的CODING账号和密码）

git fetch IFC

cd Tools/sitl\_gazebo

git remote add IFC https://e.coding.net/ifccq/aerocar/sitl\_gazebo.git（需要你的CODING账号和密码）

git fetch IFC

#### 1.3.4安装所有工具链

在根目录下打开终端，输入指令：

sudo apt install make

bash ./Aerocar/Tools/setup/ubuntu.sh

sudo apt-get install -y gcc-arm-linux-gnueabihf g++-arm-linux-gnueabihf

sudo apt install python3-pip

sudo apt install ninja-build exiftool ninja-build protobuf-compiler libeigen3-dev genromfs xmlstarlet libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev python3-pip

【提示】：上述安装如若安装过程出现问题，可以通过更改国内软件源或者科学上网，多次运行以免安装不齐全。

#### 1.3.5重启计算机

#### 1.3.6安装ROS

参照官网教程：http://wiki.ros.org/cn/Installation/Ubuntu

#### 1.3.7安装QGC

参照官网教程：

<https://docs.qgroundcontrol.com/master/en/getting_started/download_and_install.html>

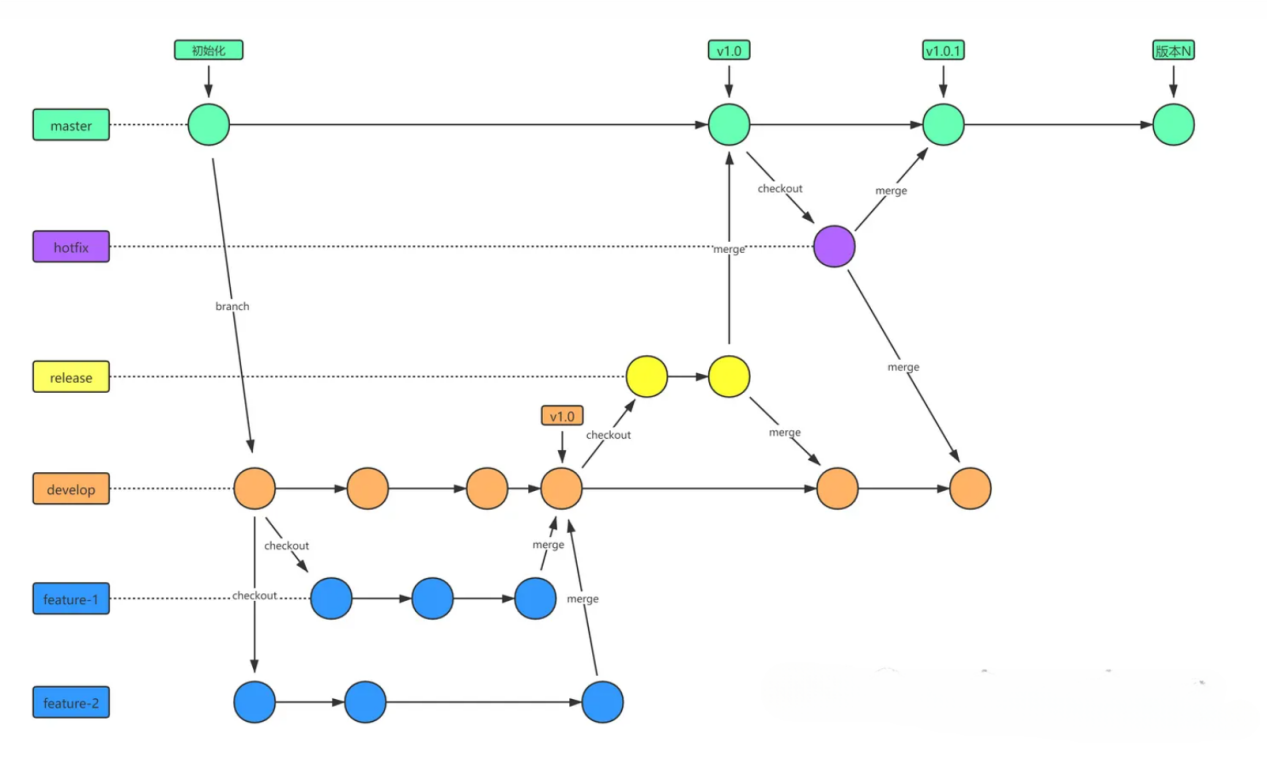
1.3.8配置仿真环境

在根目录下打开终端，输入指令：

cd AeroCar

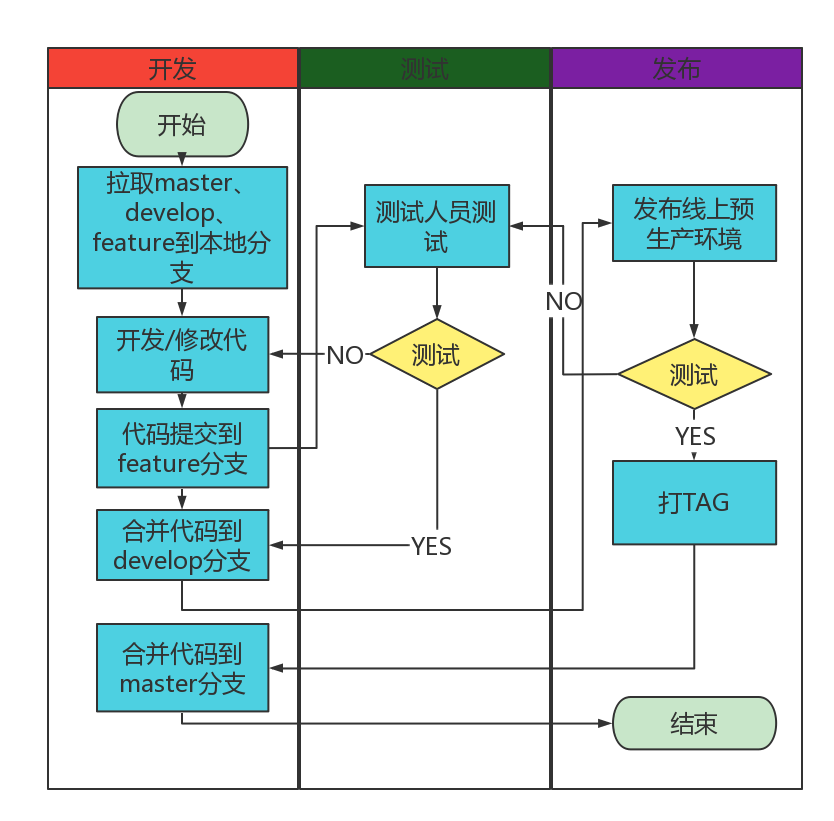
make px4\_sitl gazebo\_aerocar\_v2

## 开发流程

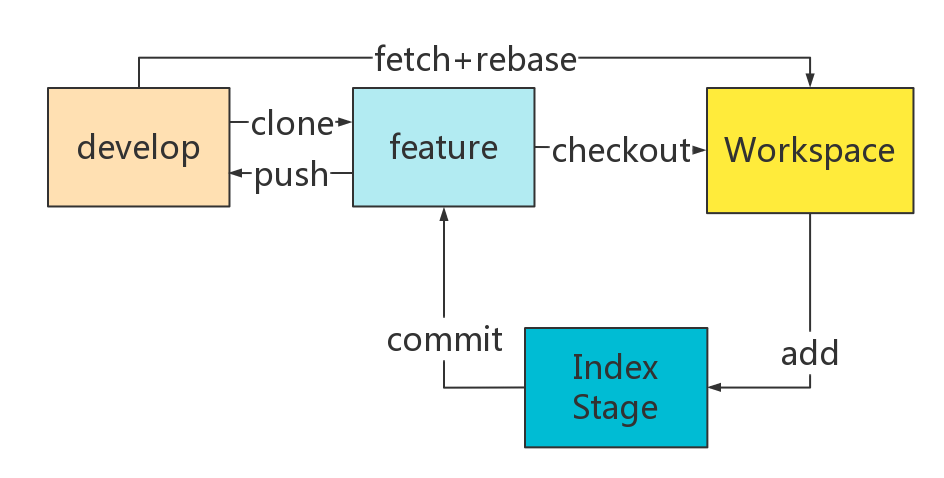


项目开发流程图

1. master分支记录了每次版本发布历史和tag标记。
2. develop分支记录了所有开发的版本历史。
3. develop分支仅第一次创建时从master分支拉取。
4. feature分支是从develop分支拉取的分支。
5. 每个feature完成后需合并到develop分支。
6. release分支是在所有功能开发自测完成后，从develop分支拉取的分支。
7. release分支一旦创建后，通常不再从develop分支拉取，该分支只做bug修复，文档生成和其他面向发布的任务。
8. release分支测试完成，达到上线标准后，需合并回master分支和develop分支。
9. hotfix分支是在线上出现bug之后，从master分支拉取的分支。
10. hotfix分支测试完成后，需合并回master分支和develop分支。



GIT开发流程图

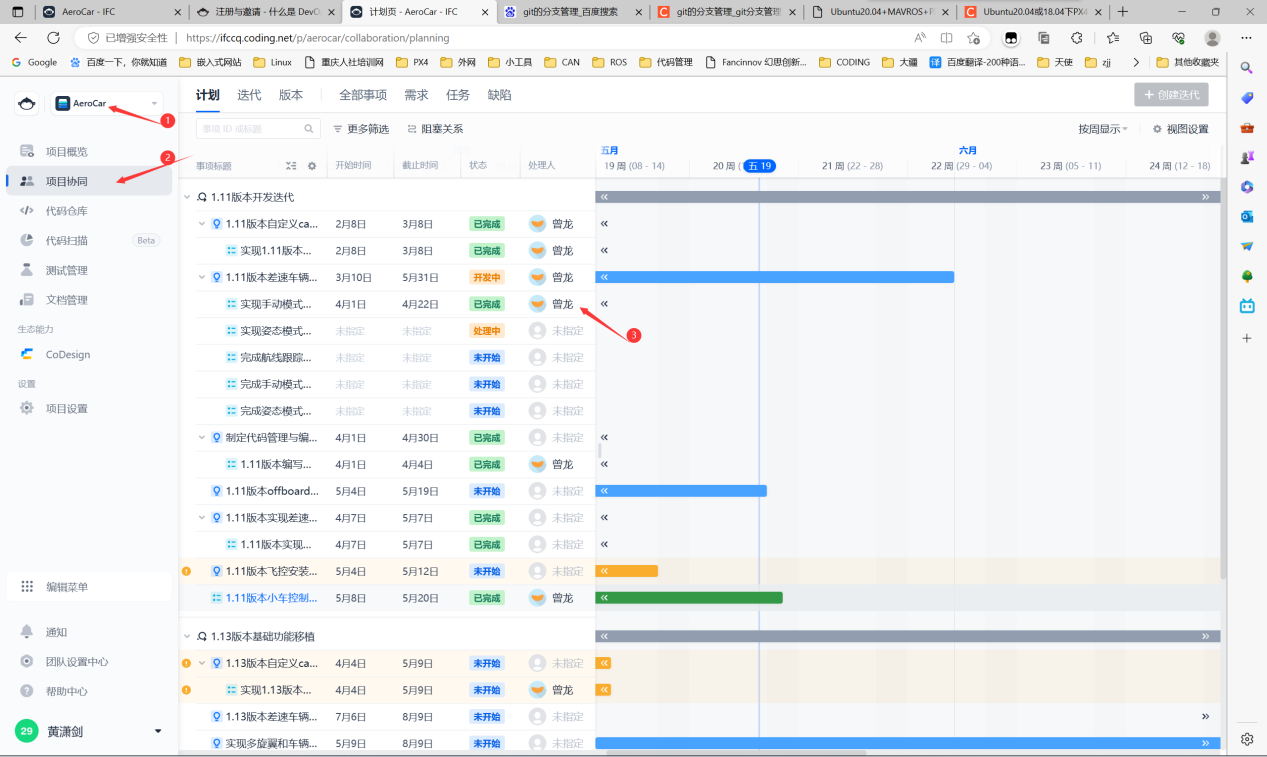


GIT操作流程图

### 2.1分支管理

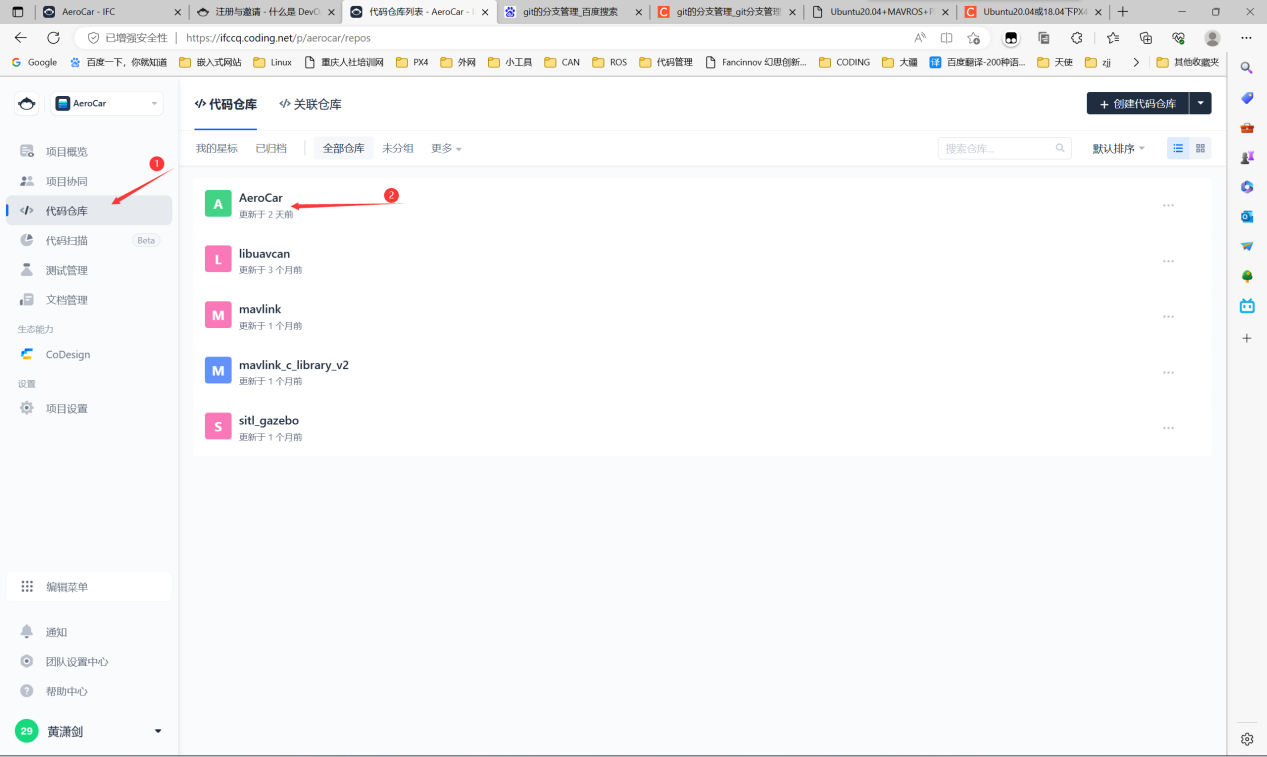
#### 2.1.1在CODING仓库查看自己的任务（feature）

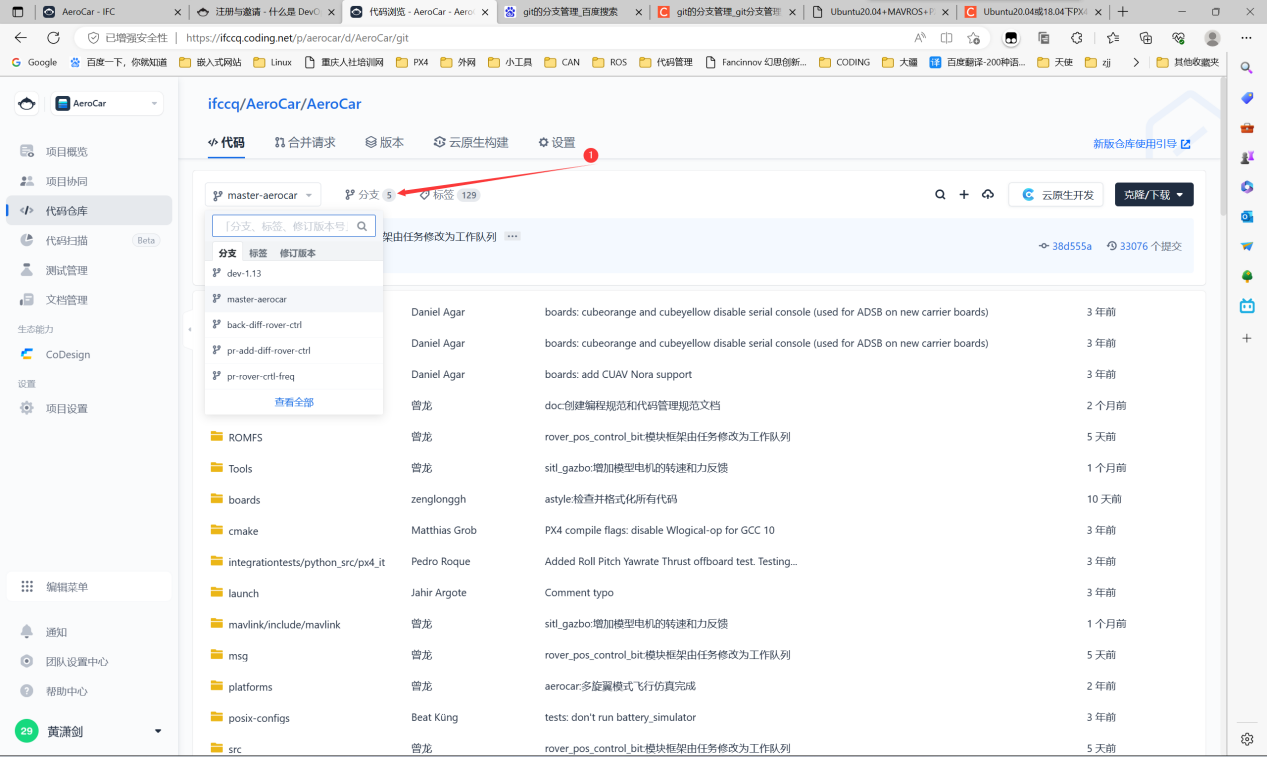
如图：

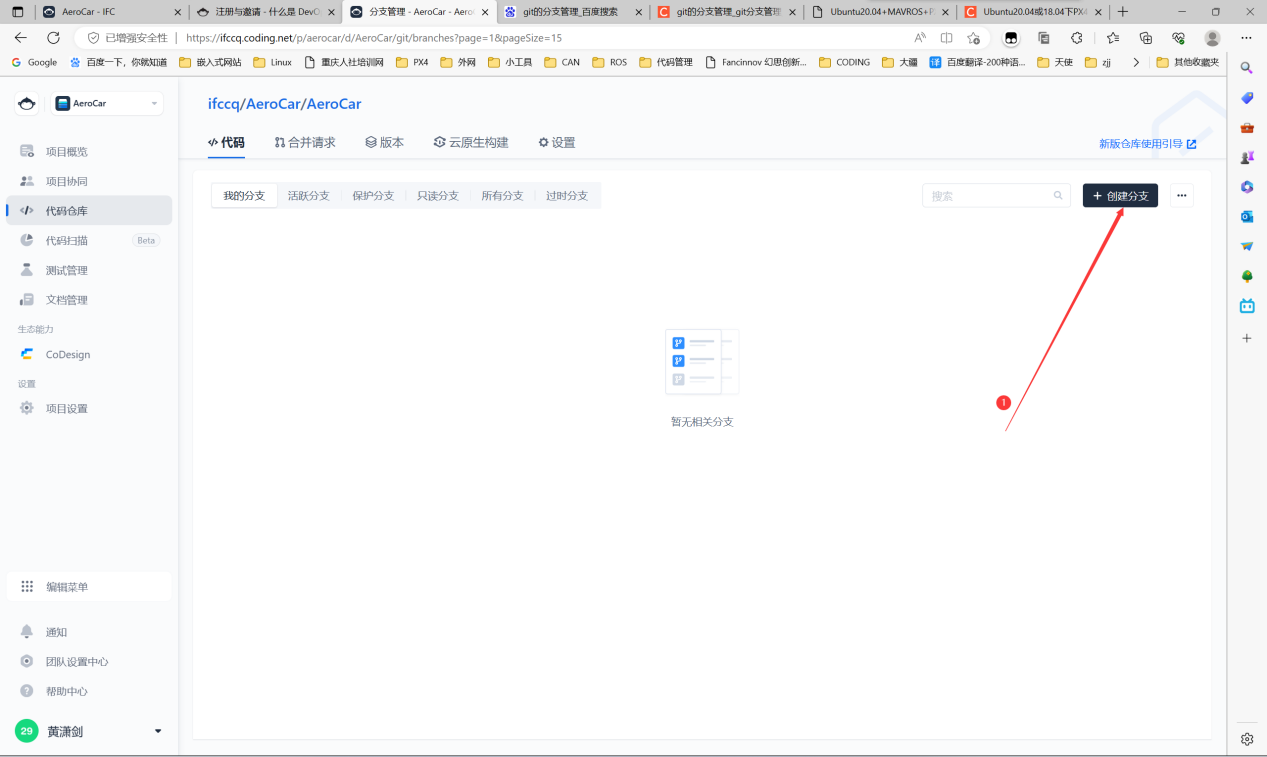


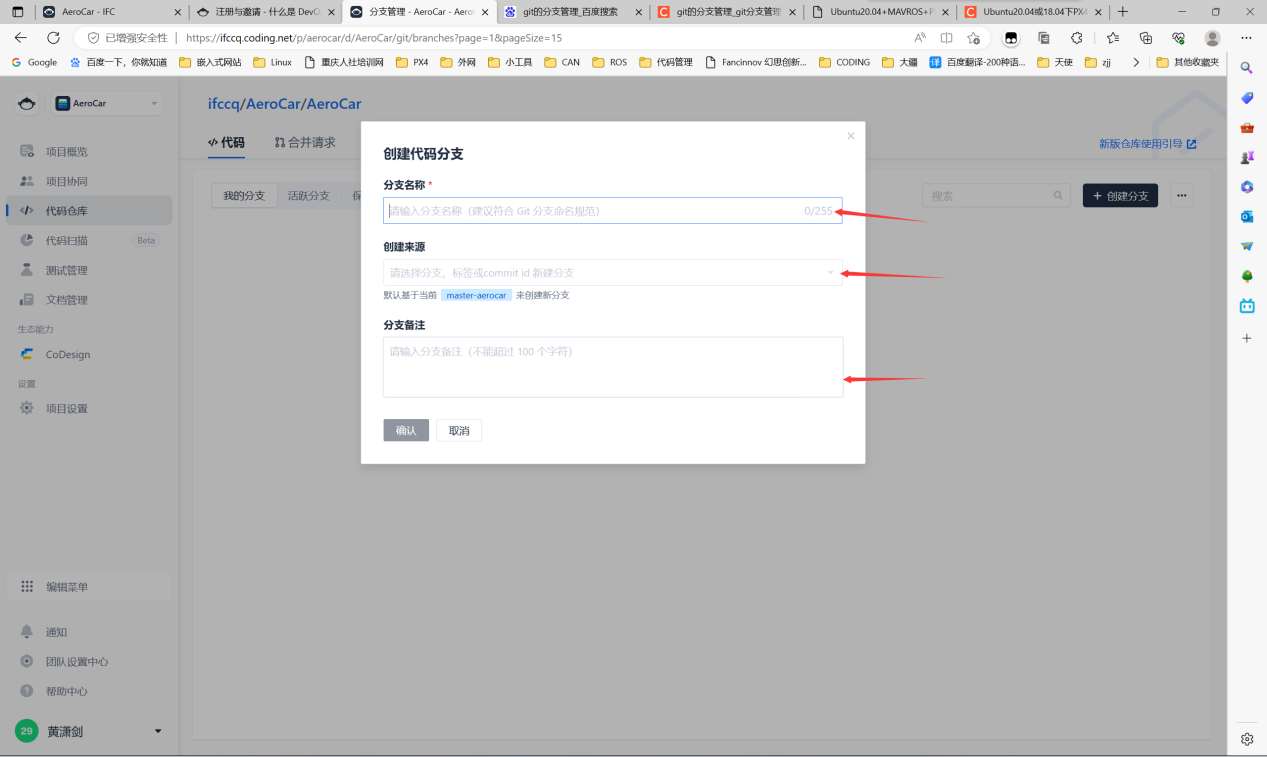
### 2.1.2根据任务创建开发分支（develop）

如图：



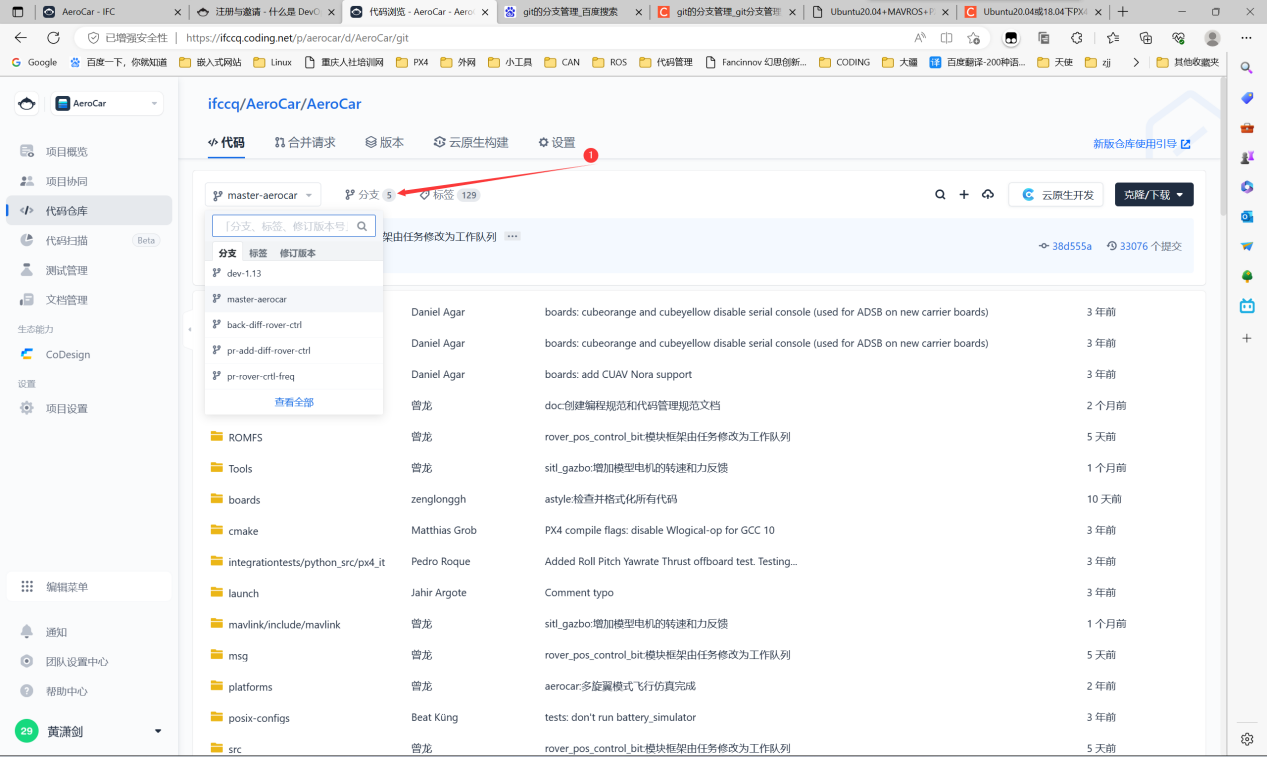


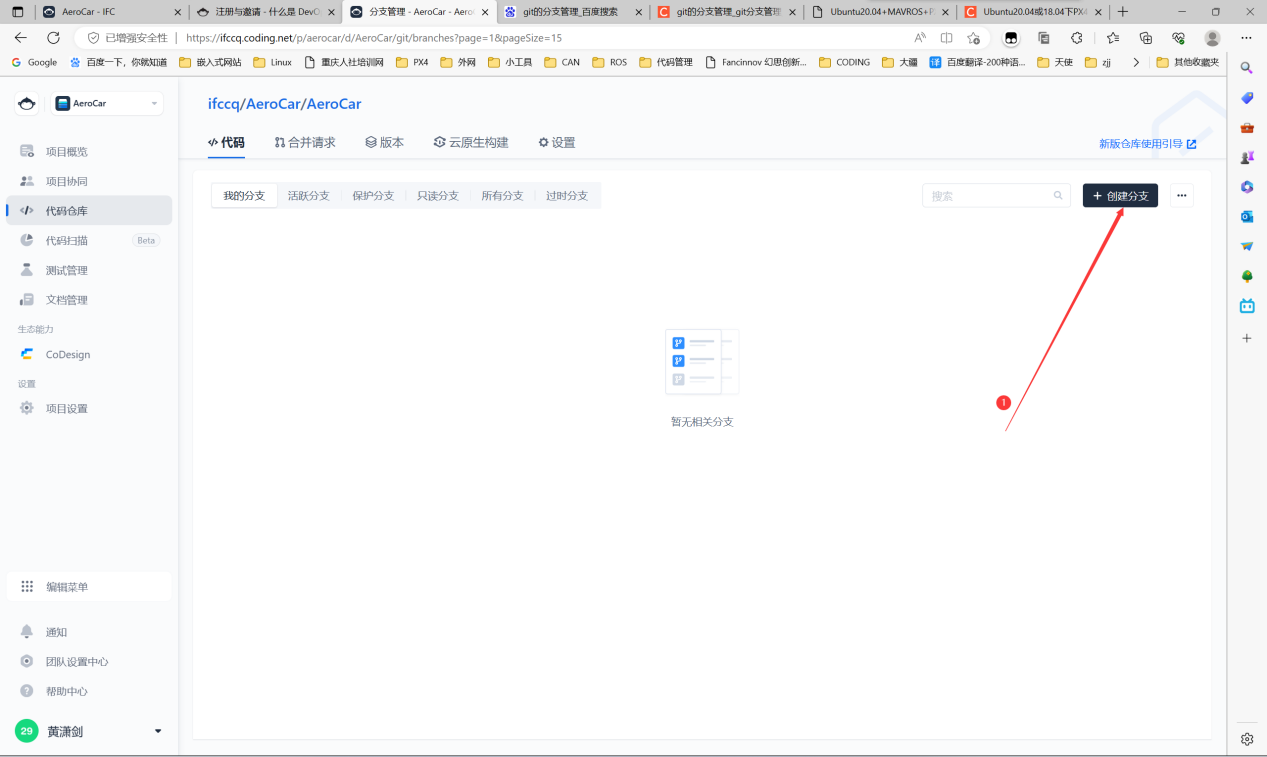


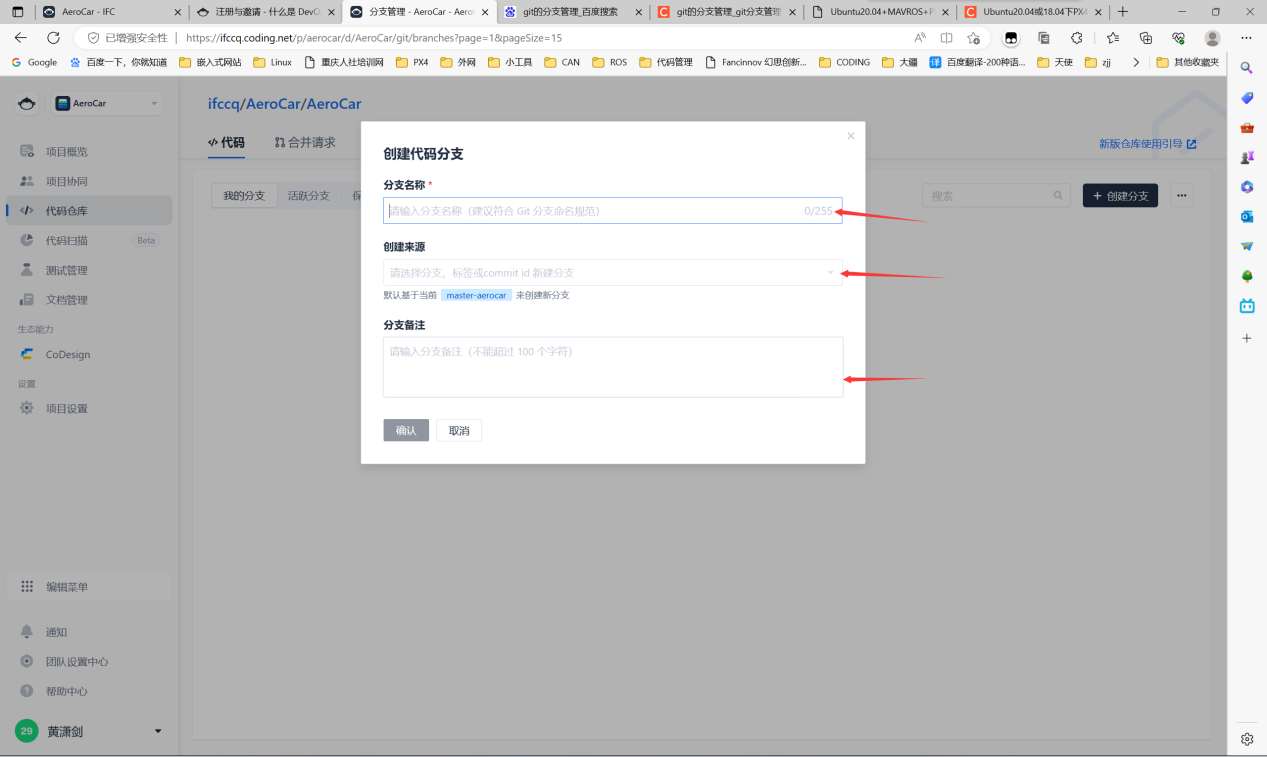


### 2.1.3根据任务创建任务分支（test\_develop）

选择你创建的开发分支，如图：







#### 2.1.4更新本地仓库（workspace）

在根目录下打开终端，输入指令：

cd AeroCar

git fetch 仓库名

#### 2.1.5切换到开发分支

在根目录下打开终端，输入指令：

cd AeroCar

git checkout 分支名

#### 2.1.6开发分支提交到CODING仓库

在根目录下打开终端，输入指令：

cd AeroCar

git add .

git commit -m “更改的内容”

git push

#### 2.1.7下拉合并主分支

开发完成后，下拉主分支然后手动合并解决冲突

git fetch

git rebase 主分支名

#### 2.1.8再次下拉合并主分支

因为可能合并的时间较长，远程又有人提交更新了，所以上一次下拉合并完成后接着再来一次，避免又有冲突的内容

git fetch

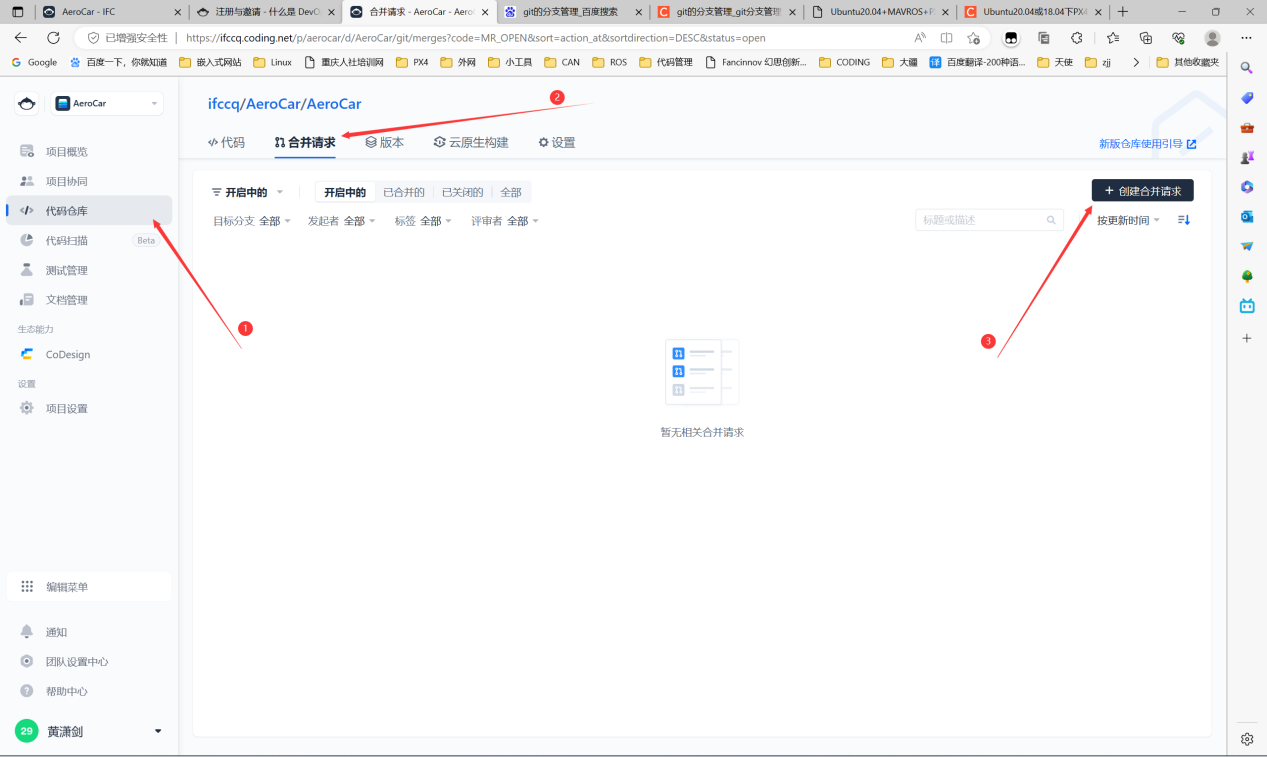
git rebase 主分支名

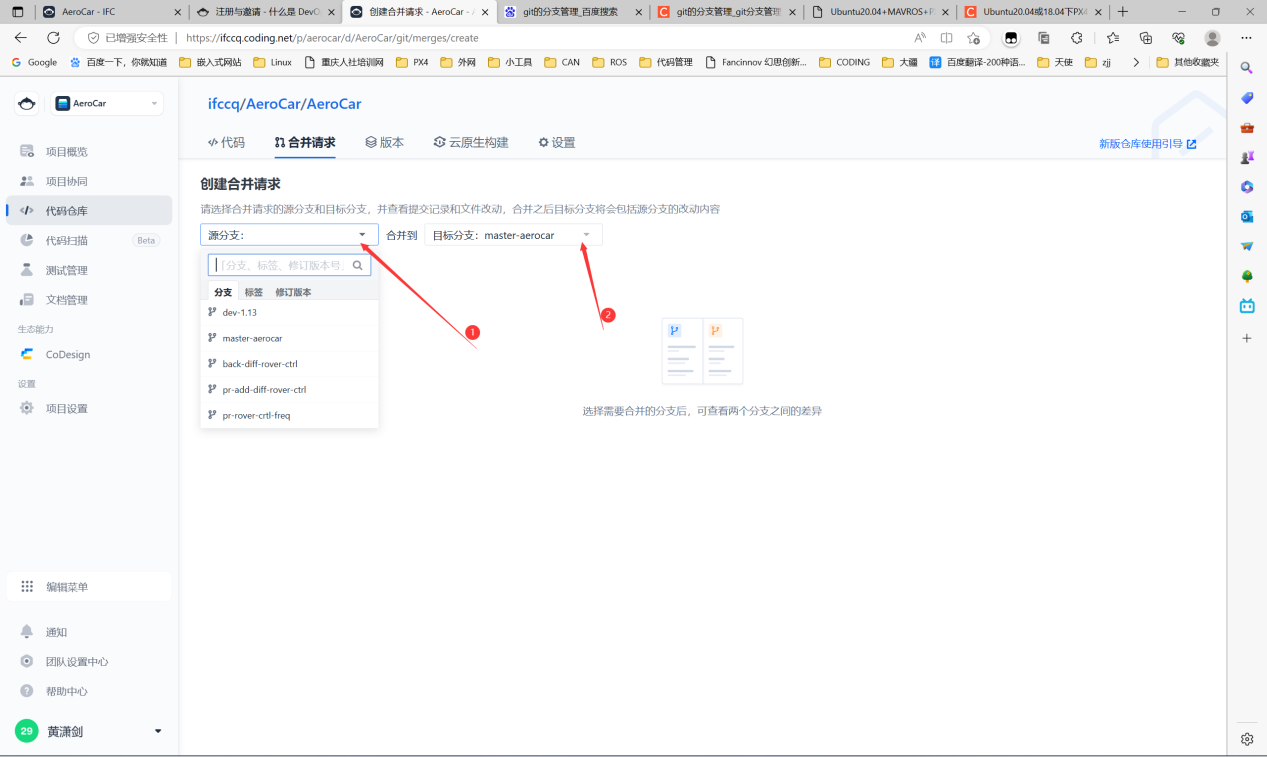
#### 2.1.9推送到CODING仓库

git push 仓库名 开发分支

#### 2.1.10团队负责人审核与合并，并发布到主分支

完成上述步骤后，按照下图在CODING上操作





### 2.2代码测试

### 2.2.1静态测试

在根目录下打开终端，输入指令：

cd AeroCar

./Tools/astyle/check\_code\_style\_all.sh

### 2.2.2仿真测试

### 2.2.3实物测试

实物测试测试项目需留存相关测试报告、数据及视频；

## 2.3下载固件

#### 2.3.1、清除编译文件

在根目录下打开终端，输入指令：

cd AeroCar

make clean

#### 2.3.2、编译文件

make cuav\_nora\_default

#### 2.3.3、烧写固件

make cuav\_nora\_default upload