**Git的起源:**

## [具体参考[链接]](https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/00137402760310626208b4f695940a49e5348b689d095fc000)

**GIT与SVN的区别:**

## [具体参考[链接]](https://www.cnblogs.com/Sungeek/p/9152223.html)

**理解GIT流:**

## [具体参考[链接]](https://guides.github.com/introduction/flow/)

**一个具体的示例(openCV参与指引):**

## [具体参考[链接]](https://github.com/opencv/opencv/wiki/How_to_contribute)

整体流程图

## **分支使用规范**

### **项目中长期存在的两个分支**

* master：主分支，负责记录上线版本的迭代，该分支代码与线上代码是完全一致的。
* develop：开发分支，该分支记录相对稳定的版本，所有的feature分支和bugfix分支都从该分支创建。

### **其它分支为短期分支，其完成功能开发之后需要删除!**

* feature/\*：特性（功能）分支，用于开发新的功能，不同的功能创建不同的功能分支，功能分支开发完成并自测通过之后，需要合并到 ****develop**** 分支，之后删除该分支。
* bugfix/\*：bug修复分支，用于修复不紧急的bug，普通bug均需要创建bugfix分支开发，开发完成自测没问题后合并到 ****develop**** 分支后，删除该分支。
* release/\*：发布分支，用于代码上线准备，该分支从****develop****分支创建，创建之后由测试同学发布到测试环境进行测试，测试过程中发现bug需要开发人员在该release分支上进行bug修复，所有bug修复完后，在上线之前，需要合并该release分支到****master****分支和****develop****分支。
* hotfix/\*：紧急bug修复分支，该分支只有在紧急情况下使用，从****master****分支创建，用于紧急修复线上bug，修复完成后，需要合并该分支到****master****分支以便上线，同时需要再合并到****develop****分支。

## **分支命名规范**

### **特性（功能）分支**

功能分支的分支名称应该为能够准确描述该功能的英文简要表述

|  |  |
| --- | --- |
| 1 | feature/分支名称 |

例如，开发的功能为 新增商品到物料库，则可以创建名称为 feature/material-add的分支。

### **bug修复分支、紧急bug修复分支**

bug修复分支的分支名称可以为项目管理工具中的bugID或者是描述该bug的英文简称

|  |  |
| --- | --- |
| 1  2 | bugfix/分支名称  hotfix/分支名称 |

比如，修复的 bug号为****MATERIAL-1****的bug，则可以创建一个名为bugfix/MATERIAL-1的分支。

### **release分支**

release分支为预发布分支，命名为本次发布的主要功能英文简称

|  |  |
| --- | --- |
| 1 | release/分支名称 |

比如，本次上线物料库新增的功能，则分支名称可以为release/material-add。

## **常用操作命令简介**

### **基本操作**

涉及到的命令主要包含以下，大家自己学习：

* git commit
* git add [–all]
* git push
* git pull
* git branch [-d]
* git merge
* git cherry-pick
* git checkout [-b] BRANCH\_NAME
* git stash

分支操作参考 [Git常用操作-分支管理](http://b.aicode.cc/git/2015/09/10/Git%E5%B8%B8%E7%94%A8%E6%93%8D%E4%BD%9C-%E5%88%86%E6%94%AF%E7%AE%A1%E7%90%86.html)

一篇非常不错的使用说明

<https://github.com/xirong/my-git/blob/master/git-workflow-tutorial.md>

需要配合完成的工作:

\* 项目列表及分支作用的列表,相关文档的整理;

\* 所有的项目代码属于公司,都属于公司的某个具体的分组,owner是公司的一个账号;

\* 所有的组成员需要有容易识别的中文名称显示,权限管理;

\* 本地的全局用户名需要设置成易识别的名称,方便日志查询;

$ git config --global user.name "杜鹤飞"

git config --global user.email jack@example.com

\* 所有的提交需要有相应的日志描述记录,方便备查;