# 初始配置

## 在Windows上安装Git配置选项

gitconfig--globaluser.name"richkoala"

gitconfig--globaluser.email"zhouh\_12@163.com"

gitconfig--globalcore.autocrlffalse//禁用自动转换

## 远程控制

ssh-keygen-trsa-C[zhouh\_12@163.com](mailto:zhouh_12@163.com)

githubsetting下添加ssh公钥

ssh-[zhouh\_12@163.com](mailto:zhouh_12@163.com)

# git基本工作流程

## 创建版本

cd<dir\_path>

gitinit //这个目录变成Git可以管理的仓库

## 添加文件

版本库中添加需要进行版本管理的文件

gitadd<top\_ecos>

gitstatus//查看文件状态1)Nocommitsyet2)Untrackedfiles

## 提交版本

gitcommit–m“comment”

gitcommit--amend-no-edit //修订处理覆盖到上一次的版本，不增加新的版本，注意版本ID发生变化

## 关联远程库

gitremoteaddorigin<https://github.com/richkoala/vc_ecos.git>

## 推送远程库

gitpush-uoriginmaster

避免用户名密码推送方式

gitremotermorigin

gitremoteaddoriginhttps://richkoala:zho13844883449@github.com/richkoala/arm\_ecos.git

gitpushoriginmaster

## 文件删除

gitrm<test.txt>

gitrm-r-n--cached"bin/"//-n：加上这个参数，执行命令时，是不会删除任何文件，而是展示此命令要删除的文件列表预览。

gitcommit-m"removetest.txt"

## 分支管理

gitcheckout-bdev //创建分支并跳转至新的分支'dev'

等效为一下两条指令

gitbranchdev //创建’dev’分支

gitcheckoutdev //跳转至’dev’分支

gitbranch //命令会列出所有分支，当前分支前面会标一个\*号。

$gitbranch

\*dev

Master

gitmergedev //将dev分支的工作成果合并到master分支上：

Updatingd46f35e..b17d20e

Fast-forward

readme.txt|1+

1filechanged,1insertion(+)

$rm-rf.git //删除.git

## 版本回溯

gitlog--oneline //查看版本信息

gitreset–hardHEAD~<num> //回到前100个

gitreset–hardgitid //回到固定版本号码

gitreflog //查看每一次的head指针变化

通过head指针指向的ID完成向未来回溯

gitreset–hardgitid

gitpushorigin<master>-f //**强制**推送至远程回到一个固定的版本

## 单个文件回溯

gitcheckout<commit\_id>--<文件名称>

## 创建标签

gitcheckout<master> //切换到需要打标签的分支上

gittag<name> //创建新标签

gittag<name><commit\_id> //为历史版本创建标签

gittag //查看标签

gitshow<tagname> //查看标签信息

gittag-a<name>-m"<taginfo>"<commit\_id>

//还可以创建带有说明的标签，用-a指定标签名，-m指定说明文字

gitpushorigin<tagname> //推送某个标签到远程

gitpushorigin–tags //一次性推送全部尚未推送到远程的本地标签

gittag-d<name> //本地标签删除

远程标签删除（需要执行一下两个操作）

1）本地删除gittag-d<tagname>

2）远程删除gitpushorigin:refs/tags/v0.9

# Git进阶工作流程

## git忽略规则

Windows下创建. .gitignore文件，按照一下规则进行添加。

.gitignore忽略规则的优先级  
在 .gitingore 文件中，每一行指定一个忽略规则，Git检查忽略规则的时候有多个来源，它的优先级如下（由高到低）：  
1）从命令行中读取可用的忽略规则  
2）当前目录定义的规则  
3）父级目录定义的规则，依次递推  
4）$GIT\_DIR/info/exclude 文件中定义的规则  
5）core.excludesfile中定义的全局规则

.gitignore忽略规则的匹配语法  
在 .gitignore 文件中，每一行的忽略规则的语法如下：  
1）**空格**不匹配任意文件，可作为分隔符，可用反斜杠转义  
2）以“**＃**”开头的行都会被 Git 忽略。即#开头的文件标识注释，可以使用反斜杠进行转义。  
3）可以使用标准的**glob**模式匹配。所谓的glob模式是指shell所使用的简化了的正则表达式。  
4）以斜杠"**/**"开头表示目录；"/"结束的模式只匹配文件夹以及在该文件夹路径下的内容，但是不匹配该文件；"/"开始的模式匹配项目跟目录；如果一个模式不包含斜杠，则它匹配相对于当前 .gitignore 文件路径的内容，如果该模式不在 .gitignore 文件中，则相对于项目根目录。  
5）以星号"**\***"通配多个字符，即匹配多个任意字符；使用两个星号"**\*\***" 表示匹配任意中间目录，比如`a/\*\*/z`可以匹配 a/z, a/b/z 或 a/b/c/z等。  
6）以问号"**?**"通配单个字符，即匹配一个任意字符；  
7）以方括号"**[]**"包含单个字符的匹配列表，即匹配任何一个列在方括号中的字符。比如[abc]表示要么匹配一个a，要么匹配一个b，要么匹配一个c；如果在方括号中使用短划线分隔两个字符，表示所有在这两个字符范围内的都可以匹配。比如[0-9]表示匹配所有0到9的数字，[a-z]表示匹配任意的小写字母）。  
8）以叹号"**!**"表示不忽略(跟踪)匹配到的文件或目录，即要忽略指定模式以外的文件或目录，可以在模式前加上惊叹号（!）取反。需要特别注意的是：**如果文件的父目录已经被前面的规则排除掉了，那么对这个文件用"!"规则是不起作用的**。也就是说"!"开头的模式表示否定，该文件将会再次被包含，如果排除了该文件的父级目录，则使用"!"也不会再次被包含。可以使用反斜杠进行转义。

示例

# 表示此为注释,将被Git忽略

\*.a表示忽略所有.a结尾的文件

!lib.a表示但lib.a除外

/TODO表示仅仅忽略项目根目录下的TODO文件，不包括subdir/TODO

build/表示忽略build/目录下的所有文件，过滤整个build文件夹；

doc/\*.txt表示会忽略doc/notes.txt但不包括doc/server/arch.txt

bin/:表示忽略当前路径下的bin文件夹，该文件夹下的所有内容都会被忽略，不忽略bin文件

/bin:表示忽略根目录下的bin文件

/\*.c:表示忽略cat.c，不忽略build/cat.c

debug/\*.obj:表示忽略debug/io.obj，不忽略debug/common/io.obj和tools/debug/io.obj

\*\*/foo:表示忽略/foo,a/foo,a/b/foo等

a/\*\*/b:表示忽略a/b,a/x/b,a/x/y/b等

!/bin/run.sh表示不忽略bin目录下的run.sh文件

\*.log:表示忽略所有.log文件

config.php:表示忽略当前路径的config.php文件

/mtk/表示过滤整个文件夹

\*.zip表示过滤所有.zip文件

/mtk/do.c表示过滤某个具体文件

被过滤掉的文件就不会出现在git仓库中（gitlab或github）了，当然本地库中还有，只是push的时候不会上传。

需要注意的是，gitignore还可以指定要将哪些文件添加到版本管理中，如下：

!\*.zip

!/mtk/one.txt

# 问题

## 推送冲突

![rejected]master->master(fetchfirst)

措施1：通过gitpull先将本地库更新到与远程库一致的版本，但要注意本地库后来做的修改可能被覆盖，最好使用gitfetch(不会自动合并)，查看更新情况再有选择合并，或者先将本地库修改过的文件备份，gitpull后再重新修改；

再运行gitpush即可成功。

措施2：git提供了一种强制上传的方式，强制将本地库上传的远程库，将会忽略版本不一致等问题，一定要谨慎使用，选项-f会用本地库覆盖掉远程库：

gitpush-f，