1.Git常用命令

1.1Git环境配置

1.1.1基本配置

1.打开Git Bash

2.设置用户信息

```
git config --global user.name "kxs"

git config --global user.email "3092734660@qq.com"
#通过上面的命令设置的信息会保存在~/.gitconfig文件中
```

查看配置信息

```
git config --global user.name
git config --list
git config --global user.email
```

1.1.2 为常用命令设置别名

1.打开用户目录，创建**.bashrc**文件

2.在**.bashrc**文件输入以下内容

```
#用户输出git提交日志
alias git-log=' git log --pretty=oneline  --all --graph --abbrev-commit'
#用于输出当前目录所有文件及基本信息
alias ll='ls -al'
```

1.2.3 解决gitbash乱码

1.打开gitbash执行下面命令

```
git config --global core.quotePath false
```

2.${git_home}/etc/bash.bashrc文件最后加入下面两行

```
export LANG="zh_CN.UTF-8"
export LC_ALL="zh_CN.UTF-8"
```

1.2 获取本地仓库

1.在电脑任意位置创建一个空目录作为我们的本地GIT仓库

2.执行git init

```
# 初始化仓库带工作区
git init
# 初始化仓库不带工作区
git init --bare
```

1.3 基础操作指令

1.3.1 查看状态 status

```
# 查看状态
git status
#查看状态 使输出信息更加简洁
git status -s
```

1.3.2 add

```
# 将未跟踪的文件加入暂存区
git add  <文件名>
# 将暂存区的文件取消暂存 (取消 add )
git reset  <文件名>

```

1.3.3 commit

```
# git commit 将暂存区的文件修改提交到本地仓库
git commit -m "日志信息"  <文件名>

```

1.3.4 克隆 clone

```
# 从远程仓库克隆
git clone 远程Git仓库地址
例如: git clone https://gitee.com/itcast/gittest.git
```

1.3.5 删除 rm

```
# 从本地工作区 删除文件
git rm <文件名>
# 如果本工作区库误删, 想要回退
git checkout head <文件名>
```

1.3.6 查看log信息

```
git log --pretty=oneline  --all --graph --abbrev-commit
```

1.3.7 版本回退

- 作用：版本切换
- 命令形式：git reset --hard commitID
    - commitID可以使用git log指令查看
- 如何查看已删除的记录
    - git reflog
    - 这个指令可以看到已经删除的提交记录

1.3.8 添加文件至忽略列表

1.创建**.gitignore**

```
#可使用通配符添加需要忽略的文件
*.a
```

1.4 分支

```
# 默认 分支名称为 master
# 列出所有本地分支
git branch
# 列出所有远程分支
git branch -r
# 列出所有本地分支和远程分支
git branch -a
# 创建分支
git branch <分支名>
# 切换分支
git checkout <分支名>
# 删除分支(如果分支已经修改过,则不允许删除)
git branch -d  <分支名>
# 强制删除分支
git branch -D  <分支名>
# 提交分支至远程仓库
git push <仓库简称> <分支名称>
# 合并分支 将其他分支合并至当前工作区
git merge <分支名称>
# 删除远程仓库分支
git push origin -d branchName
```

1.4.1 解决冲突

当2个分支修改了同一文件的同一行会出现冲突

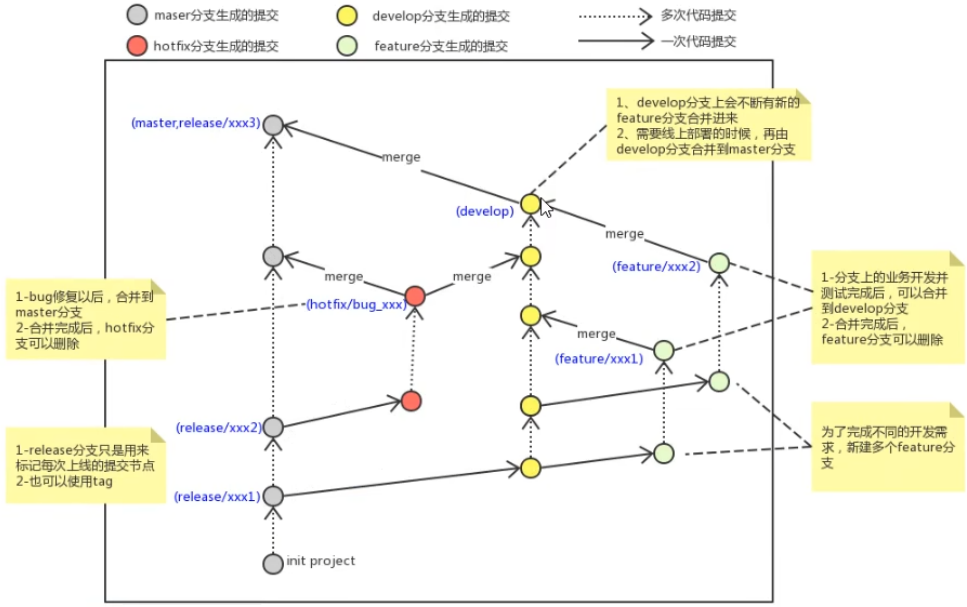
1.处理文件冲突的地方

2.将解决完冲突的文件提交

1.4.2 开发中分支使用原则与流程

几乎所有的版本控制系统都以某种形式支持分支。使用分支意味着你可以把你的工作从开发主线上分离开来进行重大的Bug修改、开发新的功能，以免影响开发主线。
在开发中，一般有如下分支使用原则与流程:

- master(生产)分支
  线上分支，主分支，中小规模项目作为线上运行的应用对应的分支;.
- develop (开发)分支
  是从master创建的分支，一般作为开发部门的主要开发分支，如果没有其他并行开发不同期上线要求，都可以在此版本进行开发，阶段开发完成后，需要是合并到master分支,准备上线。
- feature/xxxx分支
  从develop创建的分支，一般是同期并行开发，但不同期上线时创建的分支，分支上的研发任务完成后合并到develop分支。
- hotfix/xxxx分支，
  从master派生的分支，一般作为线上bug修复使用，修复完成后需要合并到master、test、develop分支。
- 还有一些其他分支，在此不再详述，例如test分支(用于代码测试)、 pre分支(预上线分支)等等。
-



2.Git远程仓库

2.1 配置SSH公钥

- 生成SSH公钥
  - ssh-keygen -t rsa
- Gitee设置账户公共钥
  - -获取公钥
    - cat ~/.ssh/id_rsa.pub

- 验证是否配置成功
- ssh -T git@gitee.com

2.2 查看远程

```
# 查看远程  列出指定的每一个远程服务器的简写
git remote
# 查看远程 ， 列出 简称和地址
git remote  -v
# 查看远程仓库详细地址
git remote show  <仓库简称>

```

2.3 添加/移除远测仓库

```
# 添加远程仓库
git remote add <shortname> <url>
# 移除远程仓库和本地仓库的关系(只是从本地移除远程仓库的关联关系，并不会真正影响到远程仓库)
git remote rm <shortname>
```

2.4 从远程仓库获取代码

```
# 从远程仓库克隆
git clone <url>
# 从远程仓库拉取 (拉取到.git 目录,不会合并到工作区,工作区发生变化)
git fetch  <shortname>  <分支名称>
# 手动合并  把某个版本的某个分支合并到当前工作区
git merge <shortname>/<分支名称>
# 从远程仓库拉取 (拉取到.git 目录,合并到工作区,工作区不发生变化) = fetch+merge
git pull  <shortname>  <分支名称>
git pull  <shortname>  <分支名称>  --allow-unrelated-histories  #  强制拉取合并
```

注意：如果当前本地仓库不是从远程仓库克隆，而是本地创建的仓库，并且仓库中存在文件，此时再从远程仓库拉取文件的时候会报错（fatal: refusing to merge unrelated histories ），解决此问题可以在git pull命令后加入参数--allow-unrelated-histories (如上 命令)

2.5 推送到远程仓库

```
# 将本地仓库推送至远程仓库的某个分支
git push [remote-name] [branch-name]
git push origin master:master
#将本地master推送到云端master并绑定
 git push --set-upstream origin master:master
 #查看绑定信息
 git branch -vv
```