

git规范及使用文档

git规范及使用文档

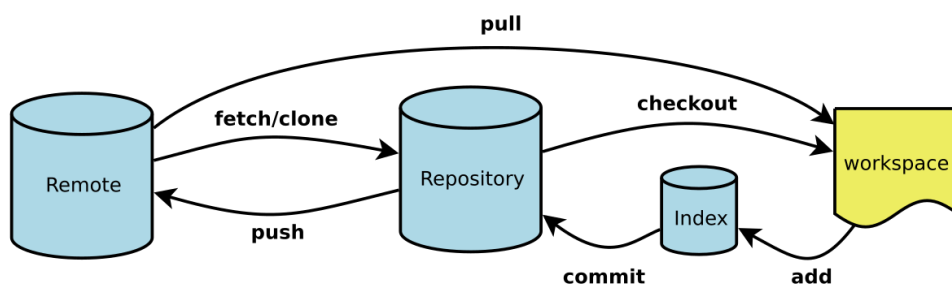
V1.0

前言

如果一个团队在使用 Git 时没有一些规范，那么将是一场难以醒来的噩梦！然而，规范固然重要，但更重要的是个人素质，在使用 Git 时需要自己养成良好的习惯。

先来看下名词解释

- Workspace: 工作区
- Index / Stage: 暂存区
- Repository: 仓库区（或本地仓库）
- Remote: 远程仓库



Git Flow

Git Flow模型中定义了主分支和辅助分支两类分支。其中主分支用于组织与软件开发、部署相关的活动；辅助分支组织为了解决特定的问题而进行的各种开发活动。

主分支

主分支是所有开发活动的核心分支。所有的开发活动产生的输出物最终都会反映到主分支的代码中。主分支分为master分支和development分支。

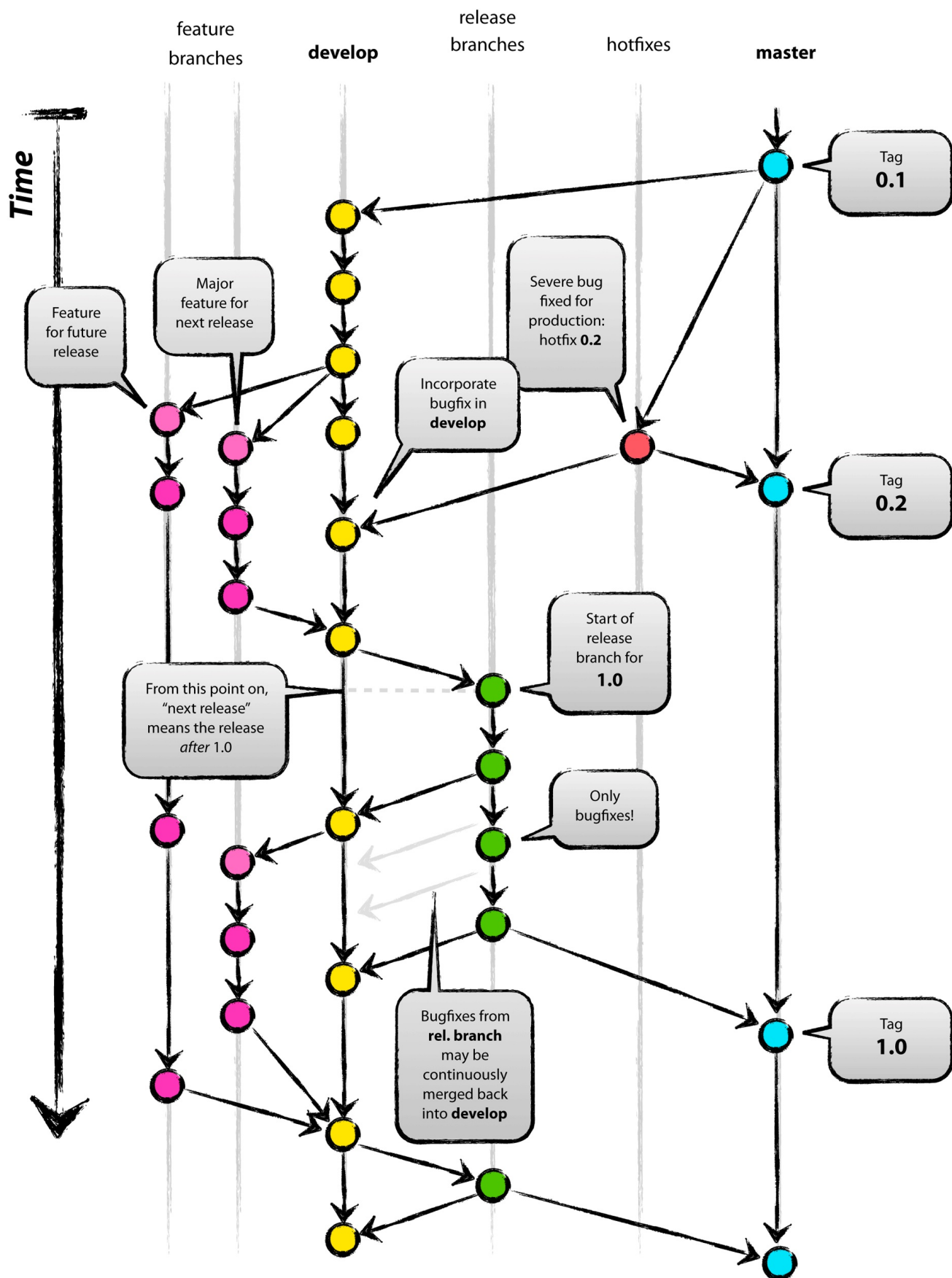
辅助分支

辅助分支是用于组织解决特定问题的各种软件开发活动的分支。辅助分支主要用于组织软件新功能的并行开发、简化新功能开发代码的跟踪、辅助完成版本发布工作以及对生产代码的缺陷进行紧急修复工作。这些分支与主分支不同，通常只会在有限的时间范围内存在。

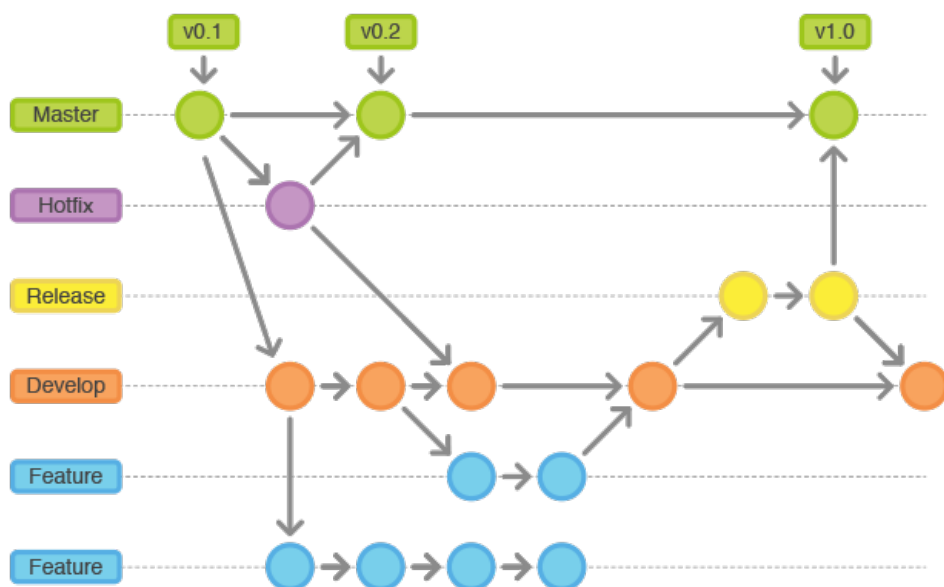
辅助分支包括：

- 用于开发新功能时所使用的feature分支；
- 用于辅助版本发布的release分支；
- 用于修正生产代码中的缺陷的hotfix分支。

以上这些分支都有固定的使用目的和分支操作限制。从单纯技术的角度说，这些分支与Git其他分支并没有什么区别，但通过命名，我们定义了使用这些分支的方法。



Git Flow的流程图



主分支

- **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。

辅助分支，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`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。

规范

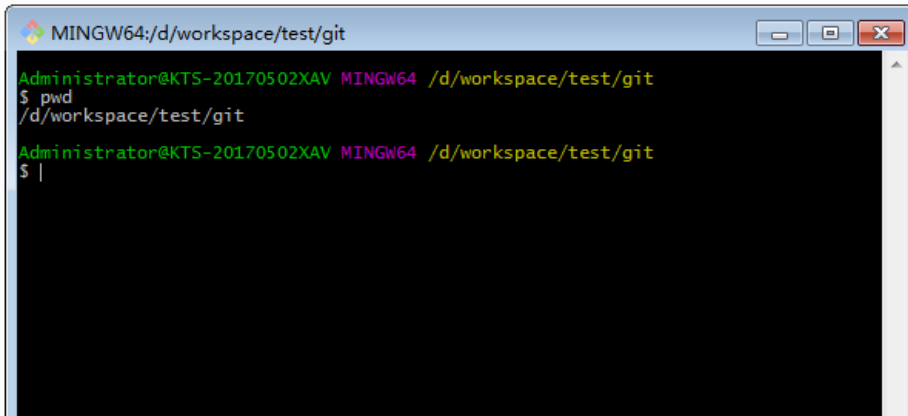
1. 提交时的粒度是一个小功能点或者一个 bug fix，这样进行恢复等的操作时能够将「误伤」减到最低；用一句简练的话写在第一行，然后空一行稍微详细阐述该提交所增加或修改的地方；不要每提交一次就推送一次，多积攒几个提交后一次性推送，这样可以避免在进行一次提交后发现代码中还有小错误。
2. 当自己一个人进行开发时，在功能完成之前不要急着创建远程分支。
3. 在将其他分支的代码合并到当前分支时，如果那个分支是当前分支的父分支，为了保持图表的可读性和可追踪性，可以考虑用 `git rebase` 来代替 `git merge`；反过来或者不是父子关系的两个分支以及互相已经 `git merge` 过的分支，就不要采用 `git rebase`了，避免出现重复的冲突和提交节点。
4. 功能开发完并自测之后，先切换到 **develop** 分支将最新的代码拉取下来，再切换回自己负责的 **feature** 分支把 **develop** 分支的代码合并进来。合并方式参照上文中的「合并」，如果有冲突则自己和配合的人一起解决。
5. 所有的新功能开发，bug修复（非紧急）都要从**develop**分支拉取新的分支进行开发，开发完成自测没有问题再合并到**develop**分支
6. **release**分支发布到测试环境，由开发人员创建**release**分支（需要测试人员提出需求）并发布到测试环境，如果测试过程中发现bug，需要开发人员**track**到该release分支修复bug，上线前需要测试人员提交**merge request**到**master**分支，准备上线，同时需要合并回**develop**分支。
7. 只有紧急情况下才允许从**master**上拉取**hotfix**分支，**hotfix**分支需要最终同时合并到**develop**和**master**分支（共两次merge操作）。

- 除了`master`和`develop`分支，其它分支在开发完成后都要删除。

使用介绍

1. 使用Git命令行

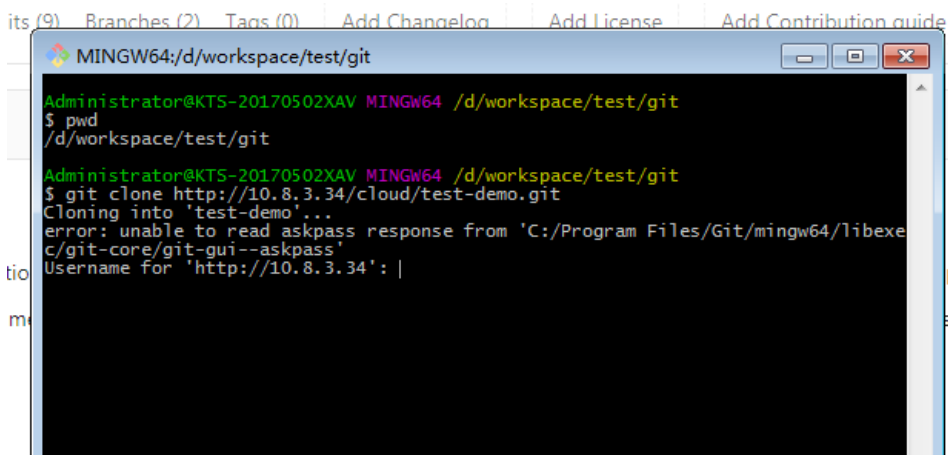
首先进入目录



```
Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git
$ pwd
/d/workspace/test/git
Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git
$ |
```

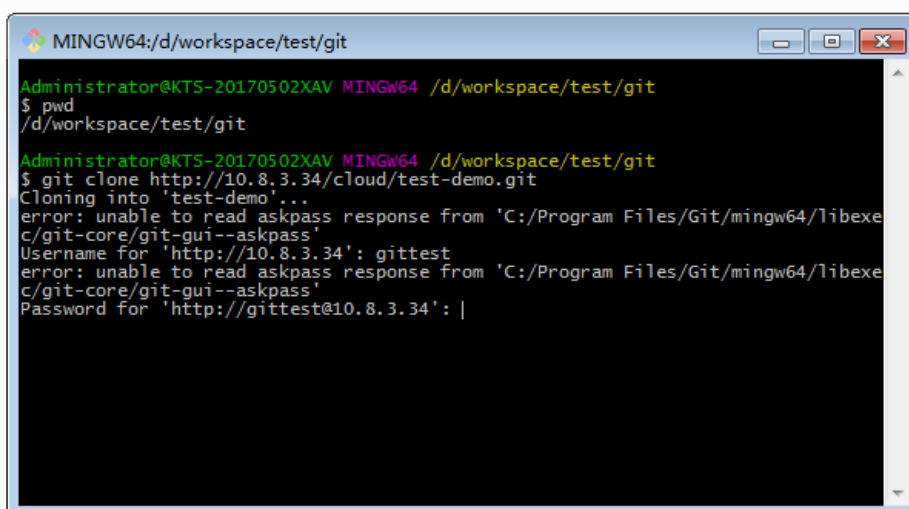
`git clone http://zzzz.git` #克隆一份代码

输入用户名



```
Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git
$ pwd
/d/workspace/test/git
Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git
$ git clone http://10.8.3.34/cloud/test-demo.git
Cloning into 'test-demo'...
error: unable to read askpass response from 'C:/Program Files/Git/mingw64/libexec
c/git-core/git-gui--askpass'
Username for 'http://10.8.3.34': |
```

输入密码

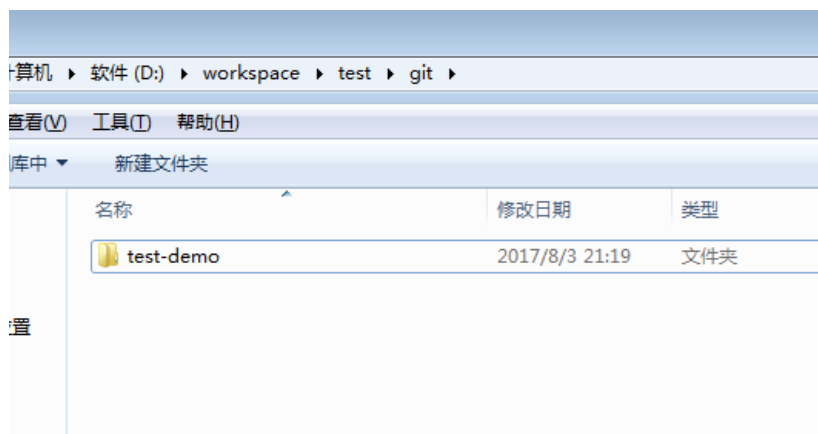


```
Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git
$ pwd
/d/workspace/test/git
Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git
$ git clone http://10.8.3.34/cloud/test-demo.git
Cloning into 'test-demo'...
error: unable to read askpass response from 'C:/Program Files/Git/mingw64/libexec
c/git-core/git-gui--askpass'
Username for 'http://10.8.3.34': gittest
error: unable to read askpass response from 'C:/Program Files/Git/mingw64/libexec
c/git-core/git-gui--askpass'
Password for 'http://gittest@10.8.3.34': |
```

代码下载完成

```
MINGW64/d/workspace/test/git
Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git
$ pwd
/d/workspace/test/git
Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git
$ git clone http://10.8.3.34/cloud/test-demo.git
Cloning into 'test-demo'...
error: unable to read askpass response from 'C:/Program Files/Git/mingw64/libexec/git-core/git-gui--askpass'
Username for 'http://10.8.3.34': gittest
error: unable to read askpass response from 'C:/Program Files/Git/mingw64/libexec/git-core/git-gui--askpass'
Password for 'http://gittest@10.8.3.34':
remote: Counting objects: 124, done.
remote: Compressing objects: 100% (85/85), done.
24 (delta 36), reused 92 (delta 20)
Receiving objects: 100% (124/124), 26.72 KiB | 0 bytes/s, done.
Resolving deltas: 100% (36/36), done.
Checking connectivity... done.
Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git
$ |
```

在目录下可以看到项目文件夹



git status #查看状态

```
MINGW64/d/workspace/test/git/test-demo
c/git-core/git-gui--askpass'
Password for 'http://gittest@10.8.3.34':
remote: Counting objects: 124, done.
remote: Compressing objects: 100% (85/85), done.
24 (delta 36), reused 92 (delta 20)
Receiving objects: 100% (124/124), 26.72 KiB | 0 bytes/s, done.
Resolving deltas: 100% (36/36), done.
Checking connectivity... done.
Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git
$ git status
fatal: Not a git repository (or any of the parent directories): .git
Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git
$ cd test-demo/
Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git/test-demo (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git/test-demo (master)
$
```

git branch -r #查看远程分支
git branch -a #查看所有分支

```
MINGW64:/d/workspace/test/git/test-demo
Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git/test-demo (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean

Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git/test-demo (master)
$ git remote
origin

Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git/test-demo (master)
$ git remote -v
origin http://10.8.3.34/cloud/test-demo.git (fetch)
origin http://10.8.3.34/cloud/test-demo.git (push)

Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git/test-demo (master)
$ git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/develop
  remotes/origin/master

Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git/test-demo (master)
$ |
```

把远程的develop分支下载到本地并创建本地分支develop分支，并且切换到develop分支：

```
MINGW64:/d/workspace/test/git/test-demo
remotes/origin/develop bbcdea9 mpi
remotes/origin/master 5797ae0 mpi

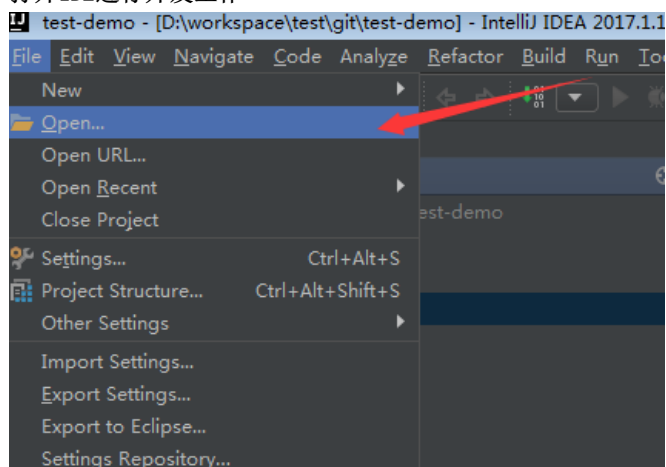
Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git/test-demo ((bbcdea9..))
$ git checkout -b develop origin/develop
Branch develop set up to track remote branch develop from origin.
Switched to a new branch 'develop'

Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git/test-demo (develop)
$ ll
total 4
-rw-r--r-- 1 Administrator 197121 1698 八月 3 13:19 pom.xml
drwxr-xr-x 1 Administrator 197121 0 八月 3 13:19 src/

Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git/test-demo (develop)
$ git branch -va
* develop          bbcdea9 mpi
  master          5797ae0 mpi
  remotes/origin/HEAD -> origin/master
  remotes/origin/develop bbcdea9 mpi
  remotes/origin/master 5797ae0 mpi

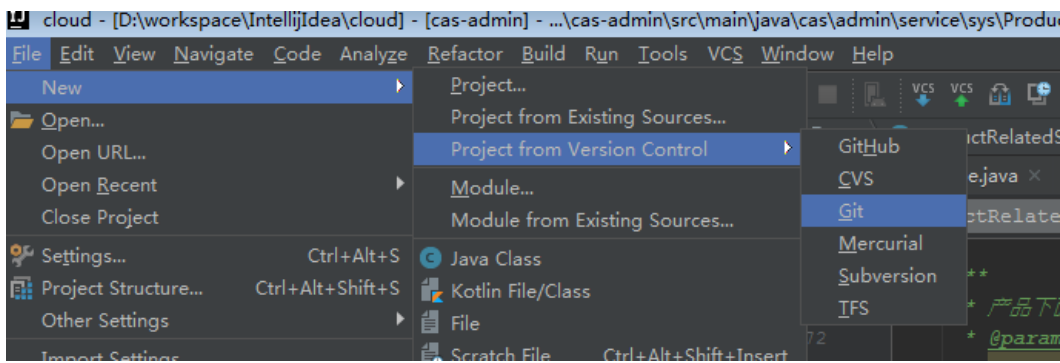
Administrator@KTS-20170502XAV MINGW64 /d/workspace/test/git/test-demo (develop)
$
```

打开IDE进行开发工作

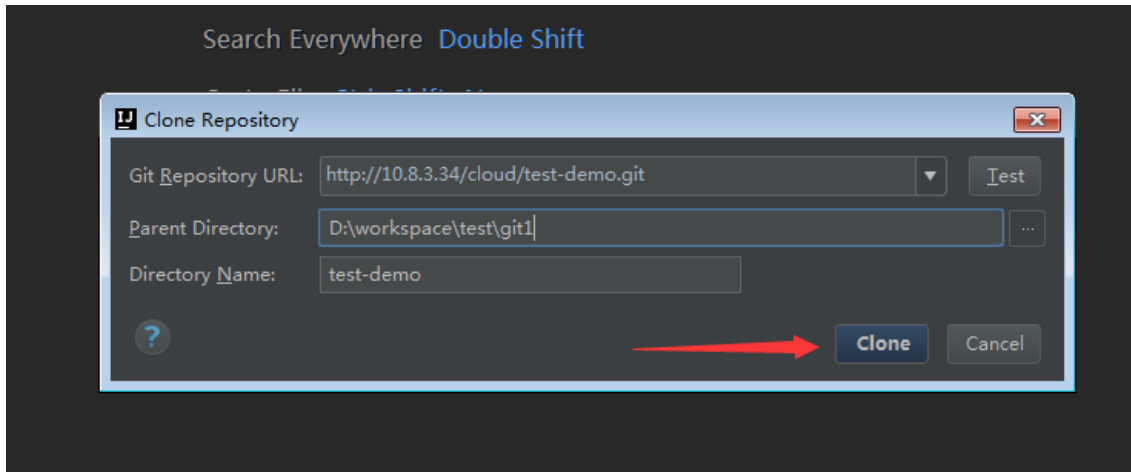


二. IDEA GIT的基本操作

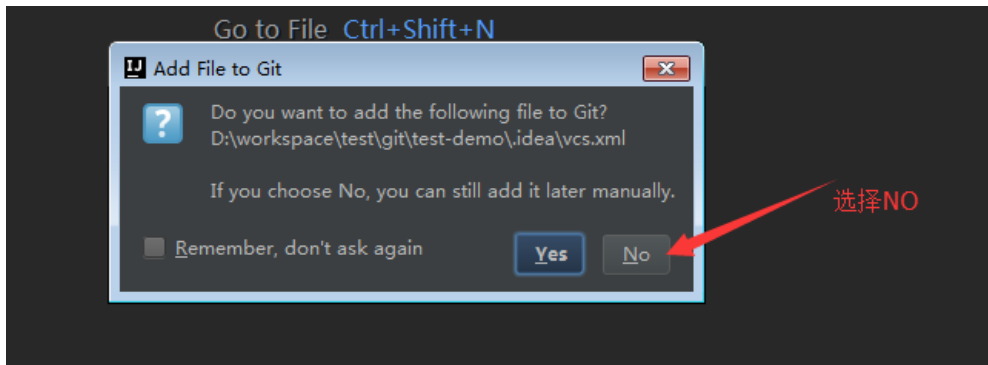
打开Project，选择从git上打开



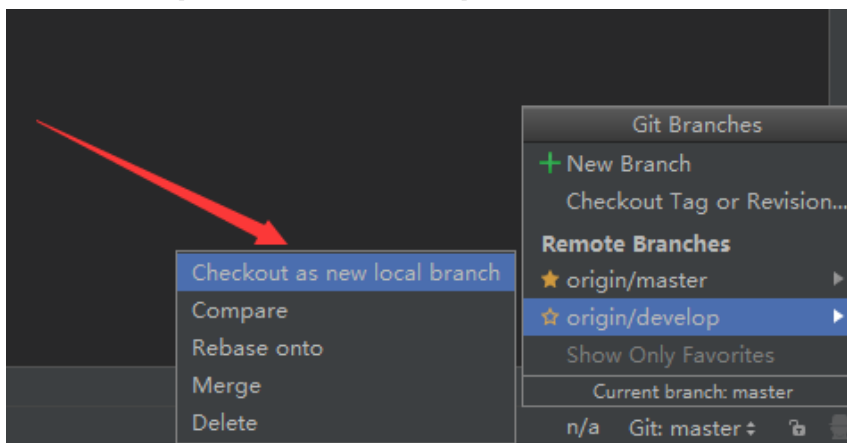
填上项目的git地址和本地的路径及项目名称



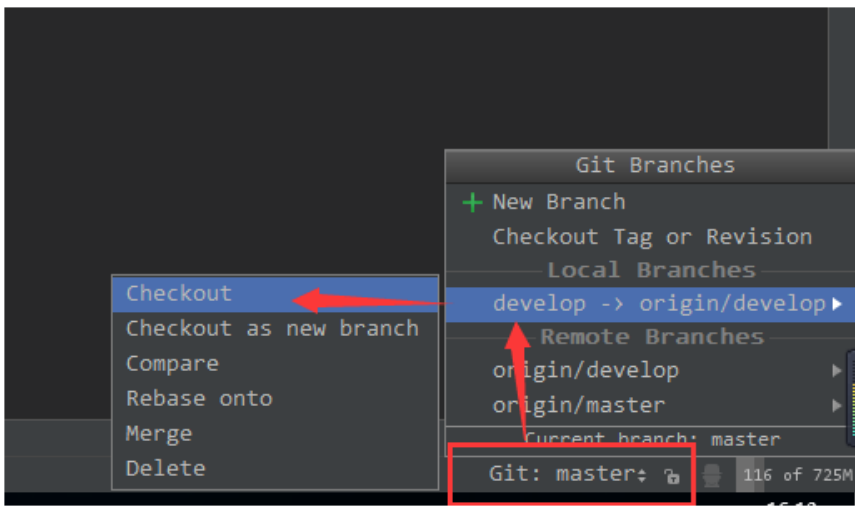
遇到类似于下面的提示，选择 “NO”



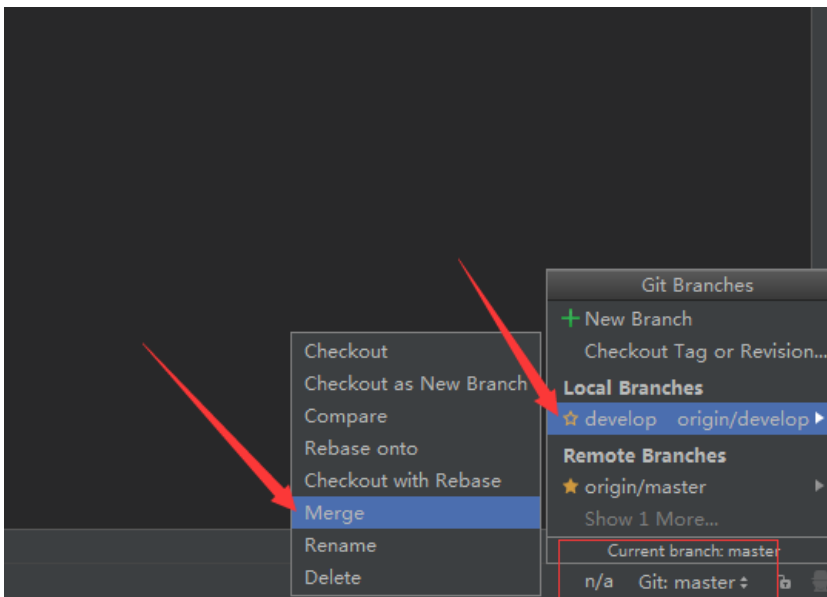
创建本地develop分支，然后切换到develop分支



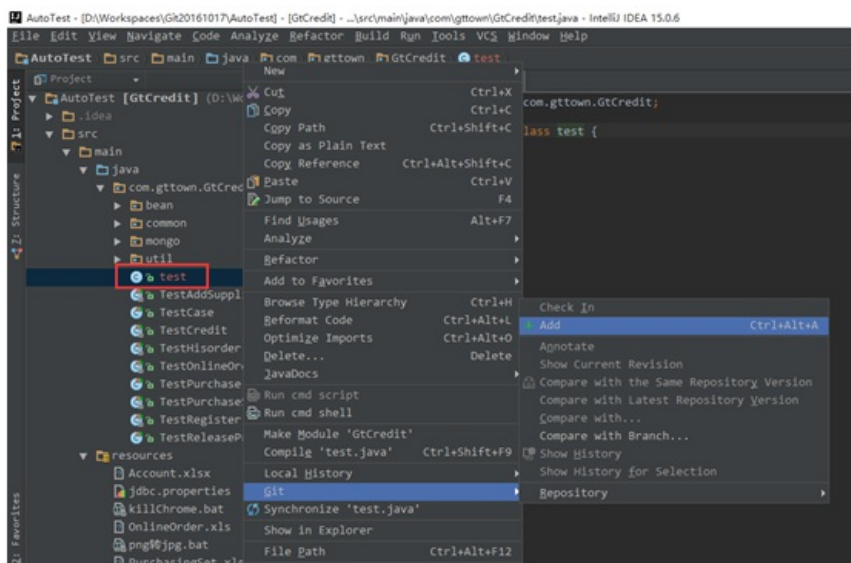
切换分支：从master切换到develop分支



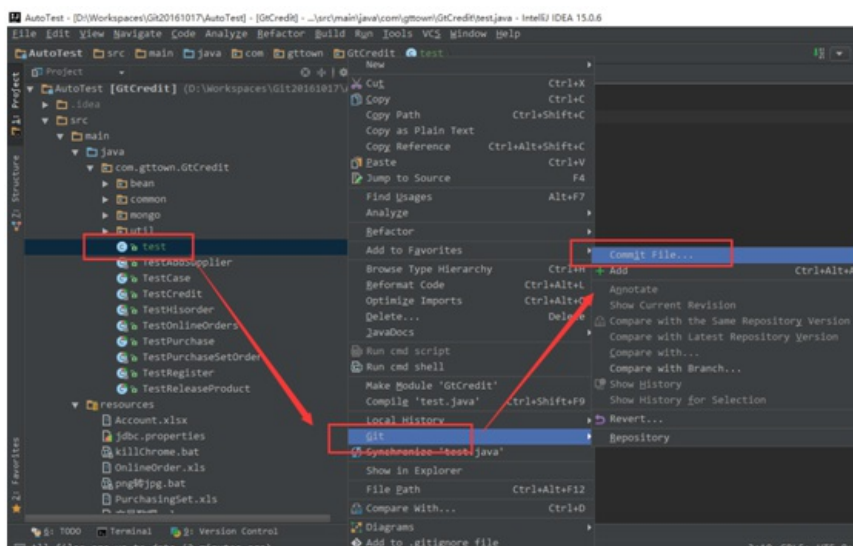
切换到master分支，合并develop分支的内容到master分支



Add文件到Git



签入到分支



PUSH到分支

