**Git的介绍**

1. git是什么？

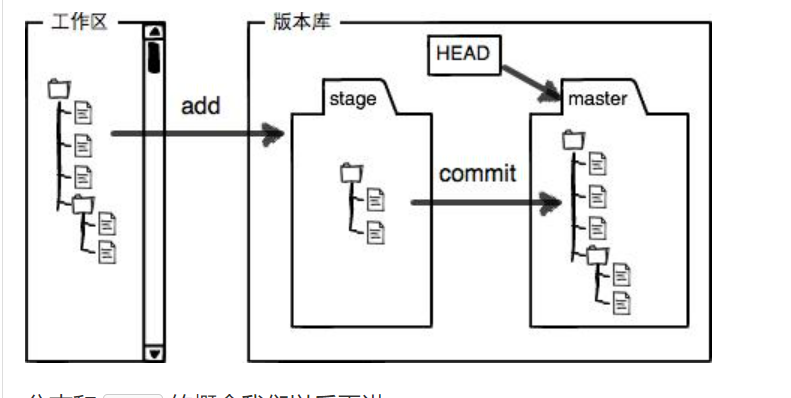
Git是一款分布式源代码管理工具(版本控制工具) 。

1. git和svn的区别？

* Git是分布式的，SVN是集中式的;
* Git复杂概念多，SVN简单易上手;
* Git分支廉价，SVN分支昂贵;

1. git的核心概念

* **工作区**(Workspace)是电脑中实际的目录。
* **暂存区**(Index)类似于缓存区域，临时保存你的改动。
* **仓库区**(Repository)，分为本地仓库和远程仓库。



1、Gitflow是什么？

Gitflow是另一种Git分支模型，它使用特性分支和多个主分支。它最初是由nvie的文森特·德里森(Vincent Driessen)出版并流行起来的。与基于主干的开发相比，Gitflow有许多更长的分支和更大的提交量。在这个模型下，开发人员创建一个特性分支，并延迟将其合并到主干分支，直到该特性完成。这些长期存在的特性分支需要更多的协作来合并，并且有更高的偏离主干分支的风险。它们还可能引入相互冲突的更新。

Gitflow可以用于有计划发布周期的项目和持续交付的DevOps最佳实践。除了特性分支工作流所需要的之外，这个工作流没有添加任何新的概念或命令。相反，它将非常具体的角色分配给不同的分支，并定义它们应该如何以及何时进行交互。除了特性分支之外，它还使用单独的分支来准备、维护和记录发布。当然，您还可以利用功能分支工作流的所有好处:拉取请求、隔离实验和更有效的协作。

2、角色分支

1.feature分支（功能分支）

2.develop分支（开发分支）

3.release分支（发布版本分支）

4.hotfixes分支（紧急版本分支、紧急修复分支）

5.master分支（主分支、核心分支）

**Git基本命令**

一、本地配置及提交

1、配置

$ git config --global user.name "Your Name"$

$ git config --global user.email "email@example.com"

1. 创建版本库

$ git init

1. 提交到版本库

$ git add .

$ git commit -m “提交的说明”

1. 推送及拉取

$ git push origin <branch-name>

$ git pull origin <branch-name>

1. 查看提交日志

$ git log

$ git log --pretty=oneline

$ git log --pretty=oneline --abbrev-commit

二、回退

1. 版本回退

$ git re**set** --hard HEAD^

$ git reset --hard commit\_id

1. 撤销修改
2. 当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时

$ git checkout --file

1. 当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令git reset HEAD <file>，就回到了场景（1），第二步按场景（1）操作。

$ git reset HEAD <file>

1. 已经提交了不合适的修改到版本库时，想要撤销本次提交，参考6、版本回退，不过前提是没有推送到远程库
2. 远程仓库
3. 查看远程库信息

$ git remote -v

origin git@github.com:michaelliao/learn-git.git (fetch)

origin git@github.com:michaelliao/learn-git.git (push)

1. 克隆远程库

$ git clone [git@github.com:michaelliao/gitskills.git](mailto:git@github.com:michaelliao/gitskills.git)

1. 本地仓库关联远程仓库

$ git remote add origin git@github.com:michaelliao/learngit.git

1. 我们第一次推送master分支时，加上了-u参数，把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来

$ git push -u origin master

1. 删除远程库

$ git remote rm origin

四、分支管理

1. 创建分支

$ git branch <name>

1. 切换分支

$ git checkout <name>

或者

$ git switch <name>

1. 创建+切换分支

$ git checkout -b <name>

或者

$ git switch -c <name>

1. 合并分支

$ git merge <name>

1. 删除分支

$ git branch -d <name>

1. 删除远程分支(并删除追踪分支)

$ git push origin --delete <branch>

1. 删除追踪分支（不会删除远程分支）

$ git branch --delete --remotes <remote>/<branch>

1. 合并a分支某次提交到b分支

$ git cherry-pick commit\_id

1. 查看分支状态

$ git status

五、储藏

1. 储藏

$ git stash

1. 恢复储藏

用git stash pop，恢复的同时把stash内容也删了

$ git stash pop

$ git stash apply 一是用git stash apply恢复，但是恢复后，stash内容并不删除，你需要用git stash drop来删除；

$ git stash apply

你可以多次stash，恢复的时候，先用git stash list查看，然后恢复指定的stash，用命令：

$ git stash apply stash@{0}

1. 查看储藏

$ git stash list

六、查看

1. 查看命令记录

$ git reflog

1. 查看当前分支状态

$ git status

1. 查看本地分支

$ git branch

1. 查看所有分支

$ git branch -a

1. 查看储藏列表

$ git stash list

1. 查看远程库信息

$ git remote -v

1. 标签

新建标签

$ git tag <tagname>

对某次提交打标签

$ git tag <tagname> commit\_id

指定标签详细信息

$ git tag -a <tagname> -m "blablabla..."

标签不是按时间顺序列出，而是按字母排序的。可以用git show <tagname>查看标签信息

$ git show <tagname>

推送一个本地标签；

$ git push origin <tagname>

推送全部未推送过的本地标签

$ git push origin --tags

删除一个本地标签；

$ git tag -d <tagname>

删除一个远程标签；

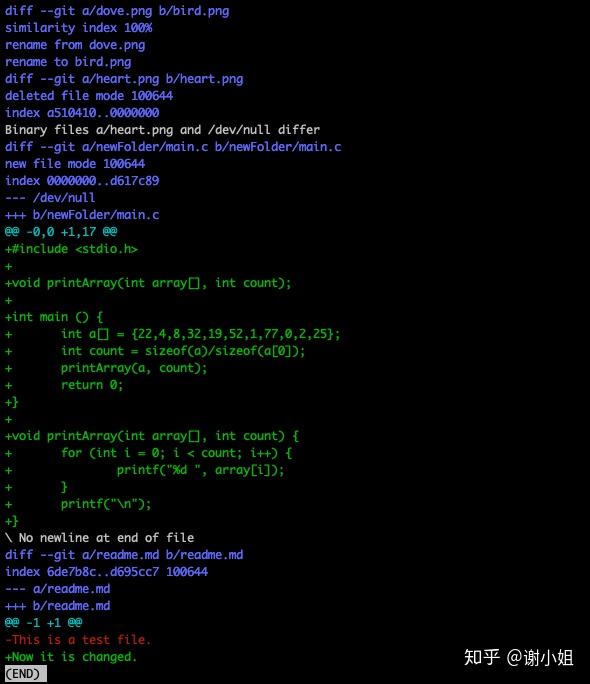
$ git push origin :refs/tags/<tagname>

1. 对比差异

$ git diff <commitA> <commitB>

带上参数 “>> fileName” 可将这些差异输出到文件 fileName 中，命令格式如下：

$ git diff <commitA> <commitB> >>fileName



我们来解读下这个差异文件。每一个变更文件作为一个段落，以 “diff --git a/file b/file” 开头，表示 a 版本的 file 文件和 b 版本的 file 文件进行比对。每一个变更文件（如果涉及到内容的修改）可以有一个或多个的变更段落，每个变更段落以 “@@ -beginline,totalline +beginline,totalline @@” 开头。其中 beginline 表示修改开始于第几行，totalline 表示这一段修改涉及的总行数。“-”表示 a 版本，“+”表示 b 版本。变更段落中绿色且以“+”开始的表示新增的内容，红色且以“-”开始的表示删除的内容。

关于 git-diff 命令的更多用法，可在官网相关页面查看。譬如可以把 commit 改为 branch 比较不同的分支，或者比较不同的两个 tag。还可以只查看某个后缀的变更，譬如只查看后缀为 .c 的文件的变更：

$ git diff <commitA> <commitB> -- \*\*/\*.c

git diff 什么参数都不加，默认比较工作区暂存区的差异

git diff --cached [<path>...]比较暂存区与最新本地版本库（本地库中最近一次commit的内容）

git diff HEAD [<path>...]比较工作区与最新本地版本库。如果HEAD指向的是master分支，那么HEAD还可以换成master

git diff commit-id [<path>...]比较工作区与指定commit-id的差异

git diff --cached [<commit-id>] [<path>...]比较暂存区与指定commit-id的差异

git diff [<commit-id>] [<commit-id>]比较两个commit-id之间的差异

git diff commit-id1 commit-id2 --stat查看两个提交版本id修改了那些文件.

git diff 版本号码1 版本号码2 src 比较两个版本的src 文件夹的差异