

版本控制系统 软件开发 Git

关注者921

被浏览229,971

为什么要先 git add 才能 git commit ?

为何不能默认之前被add的文件一定继续被加入？

关注问题

写回答

邀请回答

好问题 6

6 条评论

分享

...

64 个回答

默认排序

Ivony

1% 编程话题下的优秀答主

719 人赞同了该回答

呃，，，好像现有的答案还是没有到达那个关键点啊。

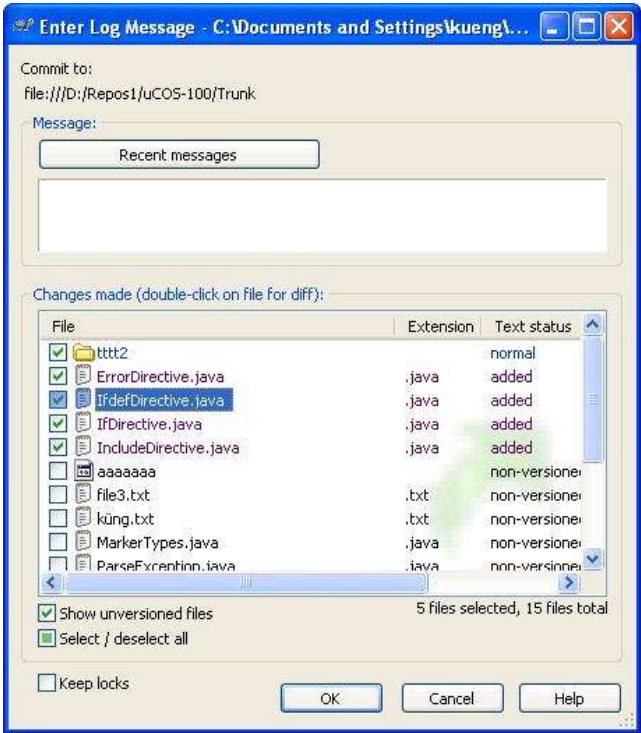
当然，最直接的原因就是git发明了一个叫做暂存区的概念，但是为什么要发明这个概念呢？没有暂存区可以不？为什么一定要有暂存区，这才是真正的问题好吧。

要解释这个问题，首先要回到GIT的前辈上来，SVN，集中式源代码管理工具的集大成者。

我们知道SVN相较于第一代源代码管理工具如VSS、CVS有几个显著的区别，其中最重要的特性之一就是原子性提交，每一个提交都是由多个文件的修改组成，而且这个提交是原子性的，要么这些修改全部成功，要么全部失败。

原子性提交带来的好处是显而易见的，这使得我们把项目整体还原到某个阶段或者时间点变得极为简便，就这一点SVN就完虐VSS等源代码管理工具。

Git作为目前逼格最高的源代码管理工具，SVN这个优良的特性显然是要借鉴的。但是Linux马上发现了一个麻烦事儿，在命令行下面选择要提交的修改，还真TM是个麻烦事儿，因为用SVN的时候我们都是这么玩的：



下载知乎客户端
与世界分享知识、经验和见解

亚马逊科技

福利来了~
虚拟专用服务器Amazon Lightsail,
750小时0元用

免费注册体验

广告

相关问题

- 如何学习好 Git ? 19 个回答
- 使用 git 时，随着版本不断的更新，以前的版本内容会不会越积越多，导致占据太多硬盘容量？ 9 个回答
- git到底怎么合并，有冲突都要手动吗？ 8 个回答
- Git频繁commit会有什么坏处吗？ 8 个回答
- 一个奇怪的git问题，求大家帮忙分析一下？ 6 个回答

相关推荐


- git 分支模型进化史**
程墨Morgan
★★★★★
- git 分支模型进化史**
程墨Morgan
★★★★★
- 狼叔：Node.js 源码是如何执行的？**
★★★★★

户还能不能愉快的玩(zhuang)耍(bi)了?

显然这有点小问题完全难不倒Linus这么一位旷世奇才。我们只需要在commit前面, 发明一个暂存区的概念就好了, 这个暂存区是可以随意的将各种文件的修改放进去的, 这样我们就可以用多个指令精确的挑选出我们需要提交的所有修改, 然后再一次性的(原子性的)提交到版本库, 问题就完美的解决了。

而且, 如果用户觉得这样实在是多此一举的话, 可以自定义一个小脚本哦, , , , 反正其实你们现在用的这些git命令, 其实大部分都是一个批处理脚本哦。

编辑于 2014-08-10



彭勇

linux/java/python/FreeSwitch/PM

365 人赞同了该回答

1. git 的 add , 是一个容易引起疑问的命令。在 subversion 中的 svn add 动作是将某个文件加入版本控制, 而 git add的意义完全不同。

继续浏览内容

 知乎

发现更大的世界

打开

 Chrome

继续

excited that staging files may soon be done via ‘git stage’ rather-than/in-addition-to ‘git add’ . This is nice for new users who often have a hard time seeing why you have to keep ‘git add’ ing to stage your changes.

事实上, 在 git 的后续版本中, 就做了两个修改:

git stage 作为 git add 的一个同义词

git diff --staged 作为 git diff --cached 的相同命令

为了容易理解, 推荐大家使用 git stage 和 git diff --staged 这两个命令, 而git add 和 git diff --cached 这两个命令, 仅仅为了保持和以前的兼容做保留。

2. 增加 stage 的带来的好处是什么?

主要有两个好处, 一个是分批、分阶段递交, 一个是进行快照, 便于回退

2.1 分批递交, 降低commit的颗粒度

比如, 你修改了 a.py, b.py, c.py, d.py, 其中 a.py 和 c.py 是一个功能相关修改, b.py, d.py属于另外一个功能相关修改。那么你就可以采用:

```
git stage a.py c.py
git commit -m "function 1"
git stage b.py d.py
git commit -m "function 2"
```

2.2 分阶段递交

比如, 你修改了文件 hello.py, 修改了一些以后, 做了 git stage heello.py动作, 相当于对当前的hello.py 做了一个快照, 然后又做了一些修改, 这时候, 如果直接采用 git commit 递交, 则只会对第一次的快照进行递交, 当前内容还保存在 working 工作区。

当前的最新修改, 则需要再做一次 git stage , 才能递交。

这中间细微的差别, 请参见:
learn.github.com/p/norm...

由于git这个特性, 需要注意到是, 每次递交之前, 需要确认是否已经将相关的修改都stage 了, 否则可能仅仅递交了部分不完整的修改。

比如你修改了部分内容, 进行了 stage, 后来你又做了一些修改, 然后就递交, 这时, 后面的修改, 并没有递交。

2.3 文件快照, 便于回退

做了部分修改以后, 进行 git s




刘看山 · 知乎指南 · 知乎协议 · 知乎隐私保护指引

应用 · 工作 · 申请开通知乎机构号

侵权举报 · 网上有害信息举报专区

京 ICP 证 110745 号

京 ICP 备 13052560 号 - 1

 京公网安备 11010802020088 号