

技术学习小组

[首页](#) / [关于我们](#) / [技术学习小组加入流程](#) / [趣编程报名流程](#) / [项目参与流程](#)

手把手教你写项目，每月仅需 599



趣编程
coolcode
可以学：
PHP、前端
Android、

输入你想了解的技术

Q SEARCH

如你对文章有问题，可联系下方『答疑人QQ』。为避免他人反感，维护学习社区，联系前请先阅读『提问的规范』



使用 Git 协作开发

[🏠](#) / [Git](#) / 使用 Git 协作开发

2016年3月13日 [👤 朱俊逸](#) [Git](#)

Git 介绍

首先从版本控制说起。版本控制是这样一种系统：它管理文件内容变化，以便将来系统成员查阅修改特定版本。有了版本控制意味着，可以把某个文件回溯到之前的状态，或者把整个项目都回退到之前某个状态，你可以比较文件的变化细节，查出最后是谁修改了哪里，又是谁在何时提交某个功能缺陷等等。你可以随意修改删除项目文件，也可以轻松恢复到原先的样子继续工作。

版本控制系统有很多种，本文只介绍 git 这种分布式版本控制。所谓分布式，类似 P2P，整个系统中无中央服务器，每个节点都有完整的代码仓库。

git 的好处主要有：

- 分布式，不依赖网络
- 开源，且简单易用
- 强大的分支管理
- 活跃的社区如 GitHub

Git 基本概念与操作

Git 绝大部分操作都在本地进行，本地有完整的仓库。这意味着 Git 可以不依赖网络，离线修改和提交代码。Git 只往数据库做数据添加，极少会进行可逆操作。一旦提交，数据就再也不会丢失。

朱俊逸



[点击这里可联系到我](#)

技术之旅

目录

- 1 Git 介绍
- 2 Git 基本概念与操作
 - 2.1 初始化
 - 2.2 文件状态
- 3 操作远程仓库
- 4 Git 协作
 - 4.1 分支
 - 4.2 利用分支开发

Git 文件有三种状态：修改，暂存和提交。对应三种工作区：工作目录，暂存区（索引）和仓库。

初始化

选择一个目录进去，将此目录变成 git 可以管理的仓库。若已有项目，只需 init 即可。

```
1 $ mkdir ~/helloworld
2
3 $ git init
4
```

或克隆现有仓库，克隆是复制 Git 仓库里所有数据，而不仅仅是工作所需的。克隆后会在当前目录下初始化，并将仓库的克隆数据存入 .git 文件夹。

```
1 $ git clone <URL>
2
```

文件状态

git 里文件有两种状态：已追踪与未追踪，已追踪即已纳入版本控制的文件。工作区所有已追踪之外的文件状态均是未追踪，可能是已修改也可能是暂存中。

检查当前的状态

```
1 $ git status
2
```

可尝试添加一个新文件，添加即把文件放入暂存区，然后查看状态

```
1 $ git add helloworld.txt
2
```

文件内容为

```
1 Hello World
2 This is my 1st git file
3
```

注意理论上 git 只能管理纯文本文件的改动，不是二进制文件也不是多媒体。为了方便管理，**请将文本文件的格式设置成各语言、各平台均通用的 UTF-8 格式。**

手把手教你写项目，每月仅需 599



趣编程
coolcode.
可以学：
PHP、前端
Android、

如你对文章有问题，可联系下方『答疑人QQ』。为避免他人反感，维护习社区，联系前请先阅读『提问的规

朱俊逸



点击这里可联系到我

技术之旅

目录

- 1 Git 介绍
- 2 Git 基本概念与操作
 - 2.1 初始化
 - 2.2 文件状态
- 3 操作远程仓库
- 4 Git 协作
 - 4.1 分支
 - 4.2 利用分支开发

然后提交更新到仓库，注意提交是指提交暂存区的文件，提交前确保文件已做 add 操作，可用 status 查看状态。

```
1 $ git commit -m "add a helloworld file"
2
```

若不满足 status 模糊的输出，可用 git diff 查看具体的修改。

git 会记录文件的每一次修改，回溯即退回任意一版本的内容。一旦对文件内容不满意或误删文件，可回溯至最近或最理想的版本，继续工作不至于丢失辛苦攒出的代码。所以每一次对文件的改动结束，一定要 commit

查看每一次的改动内容：

```
1 $ git log
2
```

如果使用 Xcode 之类的 IDE, 可用更直观的可视化图形工具查看改动。log 还可以查看具体某个作者的 commit

```
1 $ git log --author=xx
2
```

版本退回：

```
1 $ git reset --hard <指定的版本号>
2
```

向后退回版本也可以，用 reflog 查看想退回的任意一版本号。

若未 commit 的修改撤销，可直接用：

```
1 $ git checkout -- <文件名>
2
```

回滚到最近一次的 commit

操作远程仓库

```
1 $ git remote add <缩写> <url>
2 $ git fetch <缩写>
3
```

remote 命令添加远程仓库，缩写是为了方便后续操作，不用每次都输入完整 url. fetch 可拉取远程仓库中自己没有的数据。操作完成后，远

手把手教你写项目，每月仅需 599



趣编程
coolcode.
可以学：
PHP、前端
Android、

如你对文章有问题，可联系下方『答疑人QQ』。为避免他人反感，维护学习社区，联系前请先阅读『提问的规

朱俊逸



点击这里可联系到我

技术之旅

目录

- 1 Git 介绍
- 2 Git 基本概念与操作
 - 2.1 初始化
 - 2.2 文件状态
- 3 操作远程仓库
- 4 Git 协作
 - 4.1 分支
 - 4.2 利用分支开发

程仓库的数据可在本地 master 目录下访问到。

若当初本地仓库是克隆而来，源仓库默认缩写是 origin。克隆还会自动设置跟踪源仓库 master 分支，运行 pull 命令即可自动抓取与合并分支，即 pull = fetch + merge。相应的，push 也会将本地分支推送到其上游仓库。

```
1 $ git pull
2 $ git push origin master
3
```

注意，push 需要具备上游所克隆仓库的写入权限，且不能与其他人有冲突。一般无冲突很少见，需要先 fetch 他人的仓库，合并后才可成功 push。

Git 协作

分支

Git 本质上是个协作工具，意味着可以对代码仓库进行修改、同步。分支是 git 主要特性之一。分支意味着可以把个人的开发工作从主线代码中剥离，以免影响主线，与分支相对的是主线。创建一个仓库时，默认的可操作分支即主线，但实际生产时任何改动都必须在别的分支上操作，确定无问题再与主线合并。

分支的创建与删除，checkout 命令也可理解为切换到目标分支：

```
1 $ git checkout -b helloworld_x
2 $ git branch -d helloworld_x
3
```

合并分支：

```
1 $ git merge <分支名>
2
```

分支实际上只是元数据，指向原始对象的校验和，对分支的任何执行操作都非常高效。

工作场景：

1. 开发某网站。
2. 在一个新需求的分支上工作。
3. 接到报告，之前代码有个问题需要修复。

手把手教你写项目，每月仅需 599



趣编程
coolcode
可以学：
PHP、前端
Android、

如你对文章有问题，可联系下方『答疑人QQ』。为避免他人反感，维护学习社区，联系前请先阅读『提问的规

朱俊逸



点击这里可联系到我

技术之旅

目录

- 1 Git 介绍
- 2 Git 基本概念与操作
 - 2.1 初始化
 - 2.2 文件状态
- 3 操作远程仓库
- 4 Git 协作
 - 4.1 分支
 - 4.2 利用分支开发

- 4. 切换到生产分支。
- 5. 为需修复问题新建分支，完成修复。
- 6. 测试成功后，切回生产分支，合并，最后推送。
- 7. 切回最初的新需求分支，继续工作。

利用分支开发

分支管理的便捷，慢慢衍生出了一些工作流程。例如长期分支，特性分支等。

长期分支指在长时间内维持几条平行分支，定期的反复合并。几条分支不彻底合并的好处是，master 可以只保留已发布或将发布的稳定代码，而在其他平行分支上测试新功能。确保新功能稳定后，再合并入 master。实际上同时维护多个层次稳定性的分支也很常见，测试分支之下还可以有试验分支。

特性分支是种短期性的分支，可用来实现单一功能，一旦完成就删除。

Tagged: 

发表评论

电子邮件地址不会被公开。 必填项已用*标注

姓名 *

电子邮件 *

站点

发表评论

手把手教你写项目，每月仅需 599



趣编程
coolcode
可以学：
PHP、前端
Android、

如你对文章有问题，可联系下方『答疑人QQ』。为避免他人反感，维护学习社区，联系前请先阅读『提问的规范』

朱俊逸



点击这里可联系到我
技术之旅

目录

- 1 Git 介绍
- 2 Git 基本概念与操作
 - 2.1 初始化
 - 2.2 文件状态
- 3 操作远程仓库
- 4 Git 协作
 - 4.1 分支
 - 4.2 利用分支开发