

- 第1讲 Git 概述
- 课程包括
- Git概述
- 版本控制
- 版本控制工具类型
- 工作机制
- 代码托管中心

第1讲 Git 概述

课程包括

- Git
- GitHub
- Gitee
- GitLab
- TortoiseGit

Git概述


- Git 是一个免费的、开源的分布式版本控制系统，可以快速高效地处理从小型到大型的各种项目。
- Git 易于学习，占地面积小，性能极快。它具有廉价的本地库，方便的暂存区域和多个工作流分支等特性。其性能优于 Subversion、CVS、Perforce 和 ClearCase 等版本控制工具。

Git官网 <https://git-scm.com/>

- 使用的公司。



- 根据需要下载需要的版本。

 **git** --local-branching-on-the-cheap

Search entire site...

About
Documentation
Downloads
GUI Clients
Logos
Community

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read **online for free**. Dead tree versions are available on [Amazon.com](#).

Download for Windows

[Click here to download](#) the latest (2.37.1) **64-bit** version of **Git for Windows**. This is the most recent **maintained build**. It was released **27 days ago**, on 2022-07-12.

Other Git for Windows downloads

Standalone Installer

[32-bit Git for Windows Setup.](#)
[64-bit Git for Windows Setup.](#)

Portable ("thumbdrive edition")

[32-bit Git for Windows Portable.](#)

[64-bit Git for Windows Portable.](#)

Using winget tool

Install **winget tool** if you don't already have it, then type this command in command prompt or Powershell.




















```
winget install --id Git.Git -e --source winget
```

The current source code release is version **2.37.1**. If you want the newer version, you can build it from [the source code](#).

Now What?

版本控制

- 版本控制是一种记录文件内容变化，以便将来查阅特定版本修订情况的系统。
- 版本控制其实最重要的是可以记录文件修改历史记录，从而让用户能够查看历史版本，方便版本切换。

 20220628 标准版	2022/6/28 16:21	文件夹
 20220624 标准版	2022/6/24 15:51	文件夹
 20220530 标准版	2022/5/30 11:38	文件夹
 20220524 标准版	2022/5/24 16:15	文件夹
 20220520	2022/5/20 21:14	文件夹
 20220501	2022/4/24 16:48	文件夹
 20211231 标准版	2021/12/31 14:53	文件夹
 20211214 标准版	2021/12/14 15:12	文件夹
 20211213 标准版	2021/12/13 17:46	文件夹
 20211208 标准版	2021/12/9 9:16	文件夹
 20211207 标准版	2021/12/7 17:40	文件夹
 20211206 标准版	2021/12/6 16:43	文件夹
 20210907 标准版	2021/9/7 16:19	文件夹
 20210817 标准版	2021/8/17 10:35	文件夹
 20210816 标准版	2021/8/16 16:30	文件夹
 20210811 标准版	2021/8/11 15:06	文件夹
 20210802 标准版	2021/8/3 10:06	文件夹
 20210730 标准版	2021/7/30 9:57	文件夹
 20210729 标准版	2021/7/29 16:40	文件夹

- 个人开发过度到团队开发。

版本控制工具类型

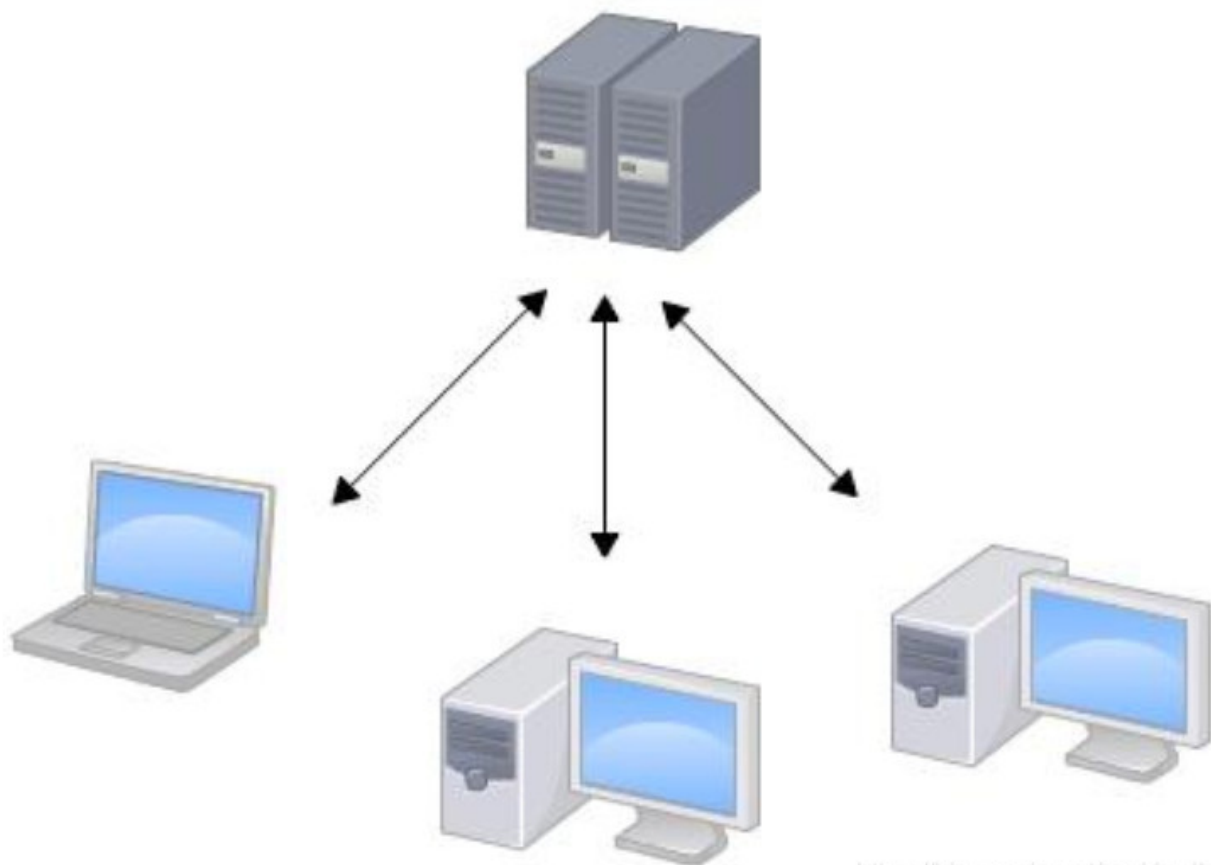
- 集中式版本工具

CVS、SVN(Subversion)、VSS.....

集中化的版本控制系统诸如 CVS、SVN 等，都有一个单一的集中管理的服务器，保存所有文件的修订版本，而协同工作的人们都通过客户端连到这台服务器，取出最新的文件或者提交更新。多年以来，这已成为版本控制系统的标准做法。

这种做法带来了许多好处，每个人都可以在一定程度上看到项目中的其他人正在做些什么。而管理员也可以轻松掌控每个开发者的权限，并且管理一个集中化的版本控制系统，要远比在各个客户端上维护本地数据库来得轻松容易。

事分两面，有好有坏。这么做显而易见的缺点是中央服务器的单点故障。如果服务器宕机一小时，那么在这一小时内，谁都无法提交更新，也就无法协同工作。



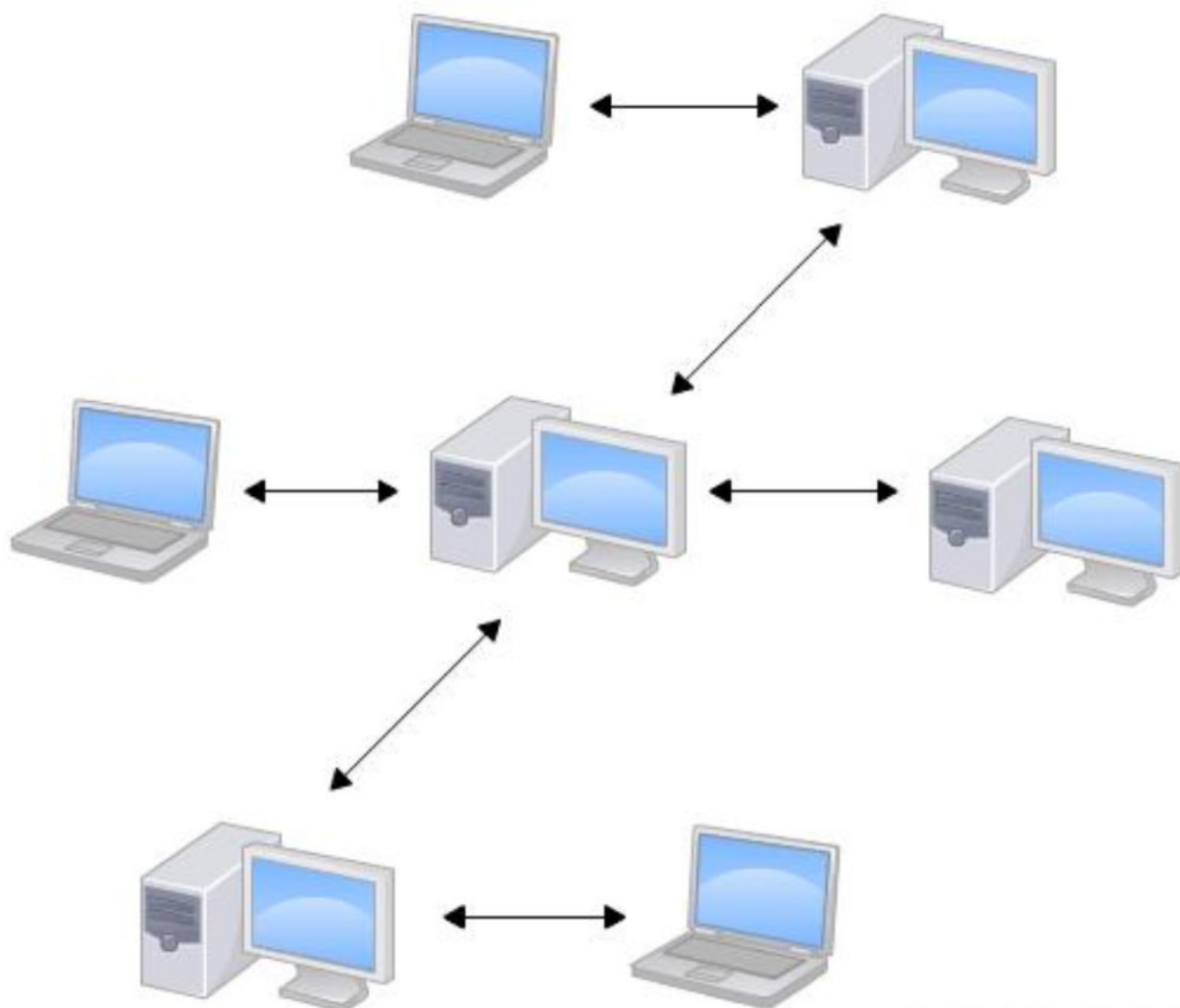
- 分布式管理控制工具

Git、Mercurial、Bazaar、Darcs.....

像 Git 这种分布式版本控制工具，客户端提取的不是最新版本的文件快照，而是把代码仓库完整地镜像下来（本地库）。这样任何一处协同工作用的文件发生故障，事后都可以用其他客户端的本地仓库进行恢复。因为每个客户端的每一次文件提取操作，实际上都是一次对整个文件仓库的完整备份。

分布式的版本控制系统出现之后,解决了集中式版本控制系统的缺陷:

1. 服务器断网的情况下也可以进行开发 (因为版本控制是在本地进行的)
2. 每个客户端保存的也都是整个完整的项目 (包含历史记录, 更加安全)



工作机制

1. 写代码 -> 工作区 `git add`
2. 临时存储 -> 暂存区 `git commit`
 - 暂存区可以删除。
3. 历史版本 -> 本地库 `git push`
 - 历史版本不可删除。
4. 推送 -> 远程库。

代码托管中心

代码托管中心是基于网络服务器的远程代码仓库，一般我们简单称为远程库。

- 局域网
 - GitLab
- 互联网
 - GitHub
 - Gitee