Git 和 GitHub 入门

讲师：张礼军

主要内容

- 版本管理的演变
- Git 基础
- GitHub 基础

一. 版本管理的演变

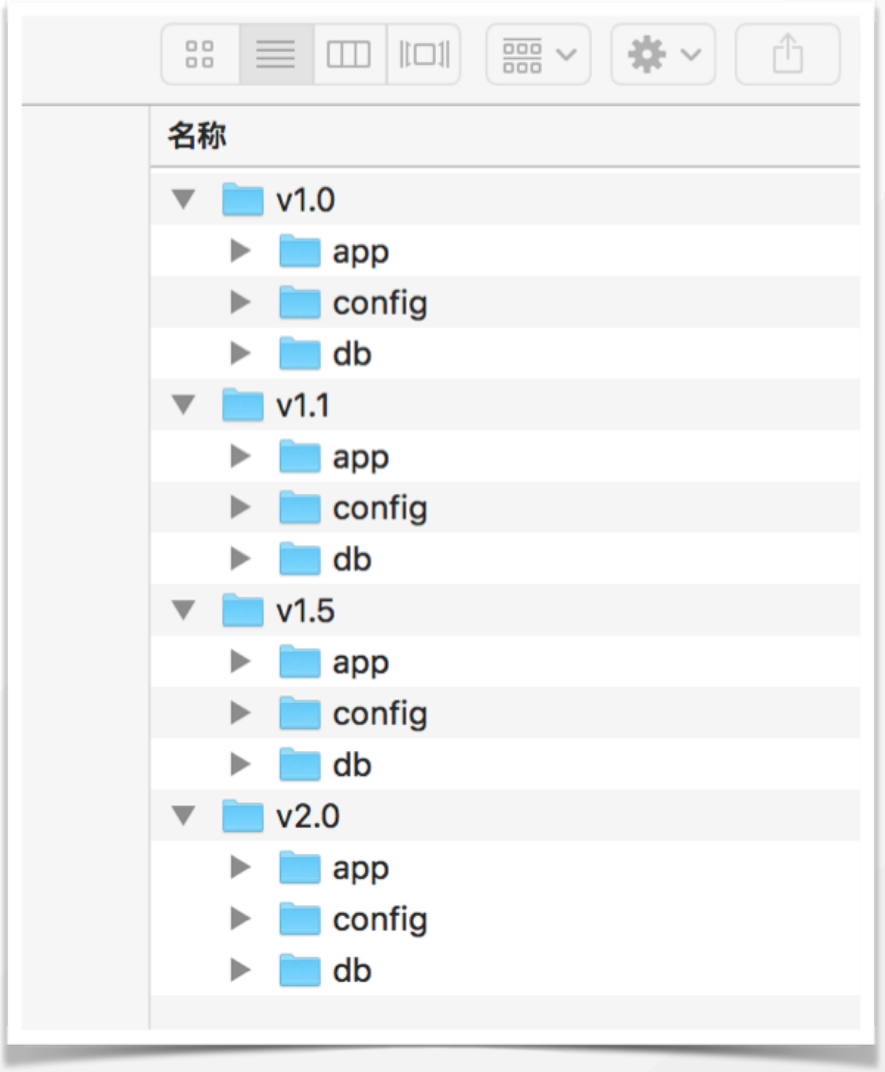
- 什么是版本控制系统？
- VCS 出现前
- 集中式 VCS
- 分布式 VCS

1.1 什么是版本控制系统？

- 版本控制系统（Version Control System），简称 VCS 。是一种记录一个或若干文件内容变化，以便将来查阅特定版本修订情况的系统。
- 版本控制系统不仅可以应用于软件源代码的文本文件，而且可以对任何类型的文件进行版本控制。用的比较多的如 git、svn 等。

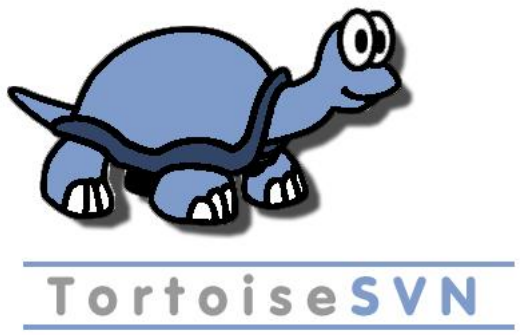
1.2 VCS 出现前

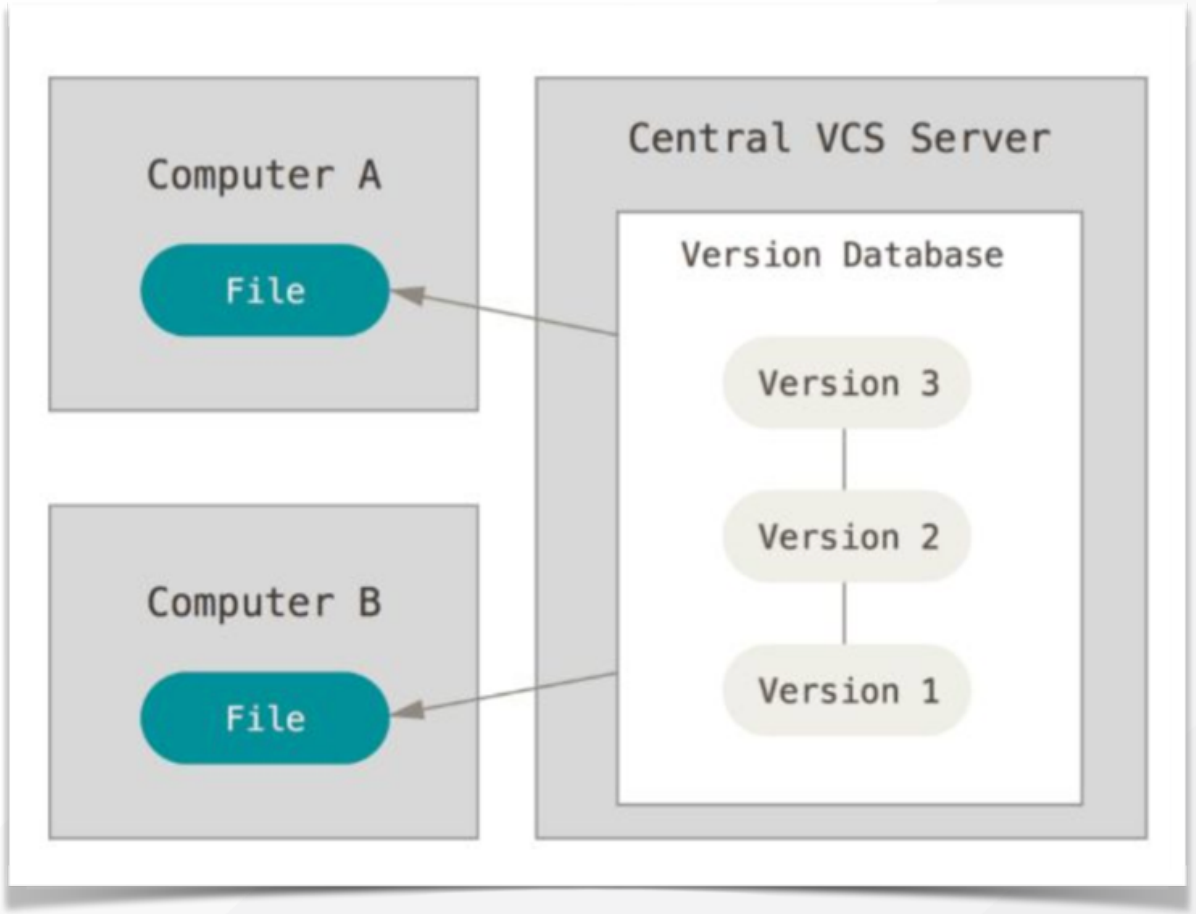
- 用目录或文件拷贝区别不同版本
- 公共文件容易被覆盖
- 成员沟通成本很高，代码集成效率低下



1.3 集中式 VCS

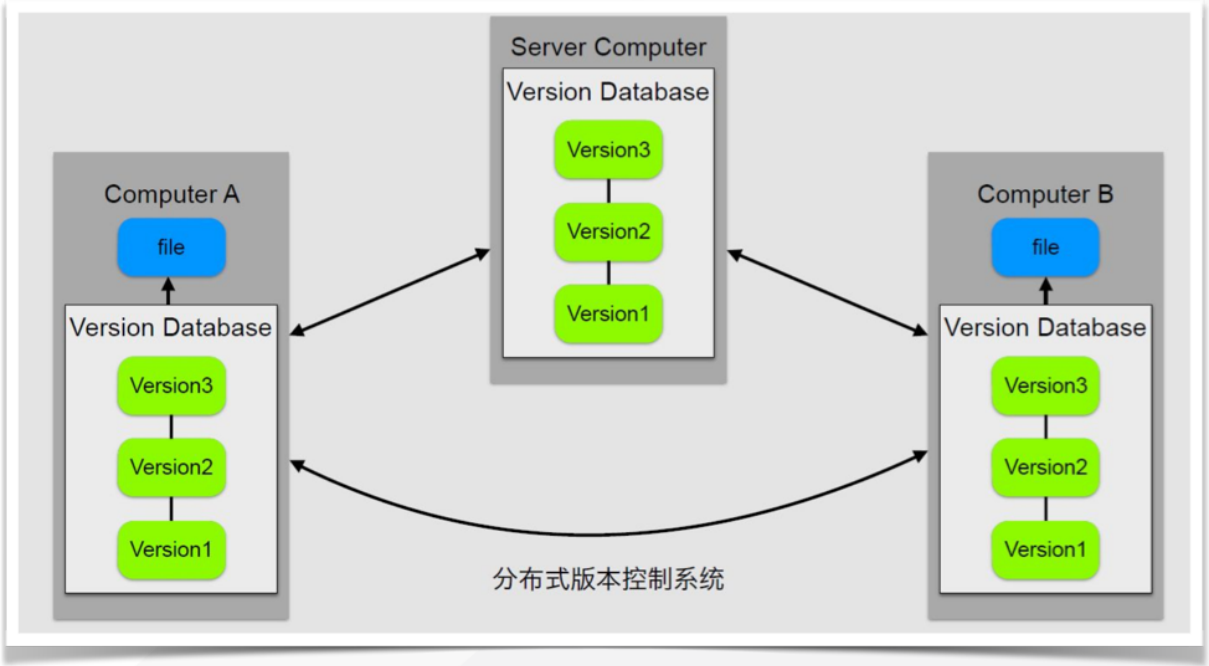
- 有集中的版本管理服务器
- 具备文件版本管理和分支管理能力
- 集成效率有明显地提高
- 客户端必须时刻和服务器相连





1.4 分布式 VCS

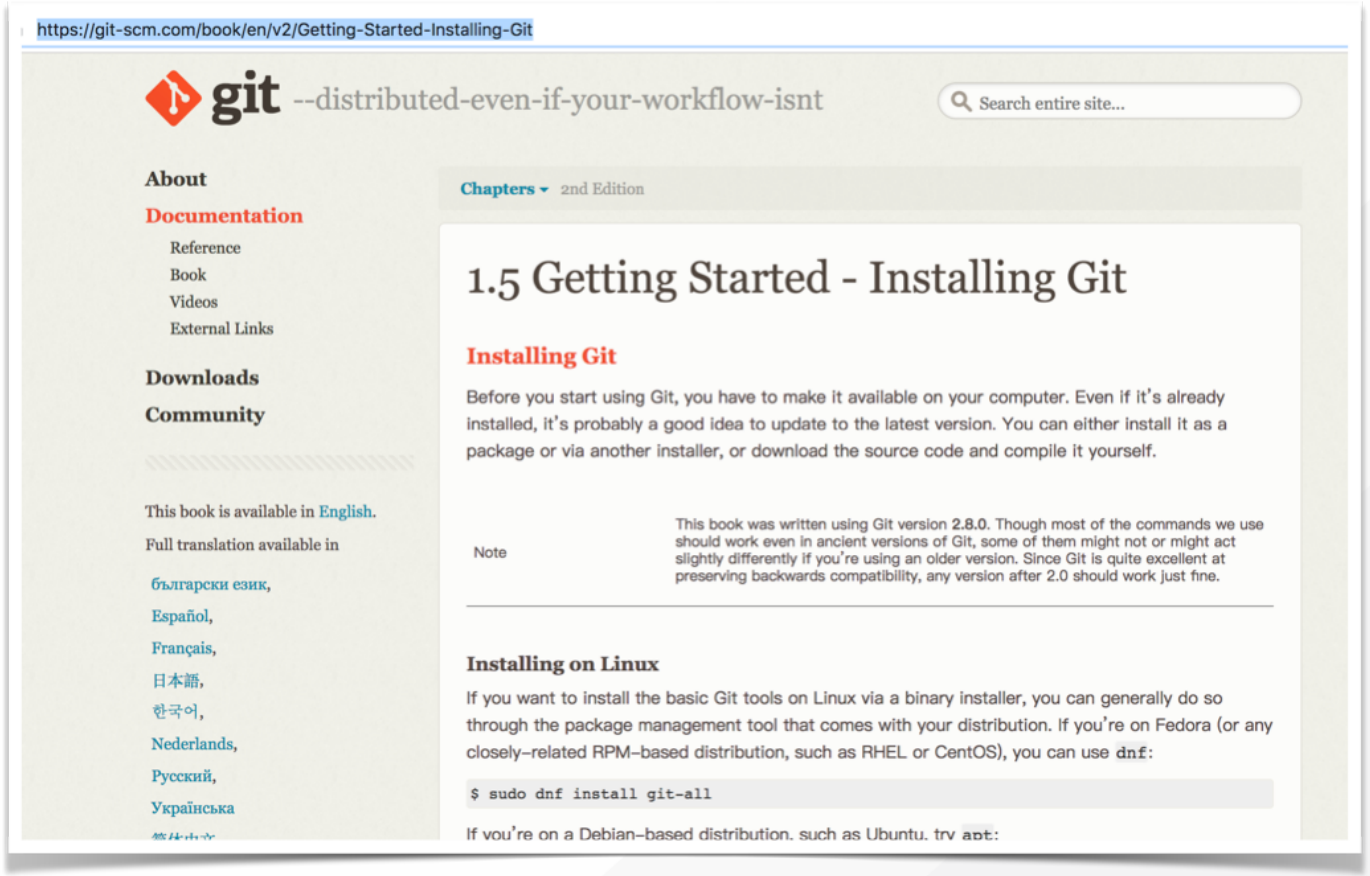
- 服务端和客户端都有完整的版本库
- 脱离服务端，客户端照样可以管理版本
- 查看历史和版本比较等多数操作，都不需要访问服务器，比集中式 VCS 更能提高版本管理效率



二. Git 基础

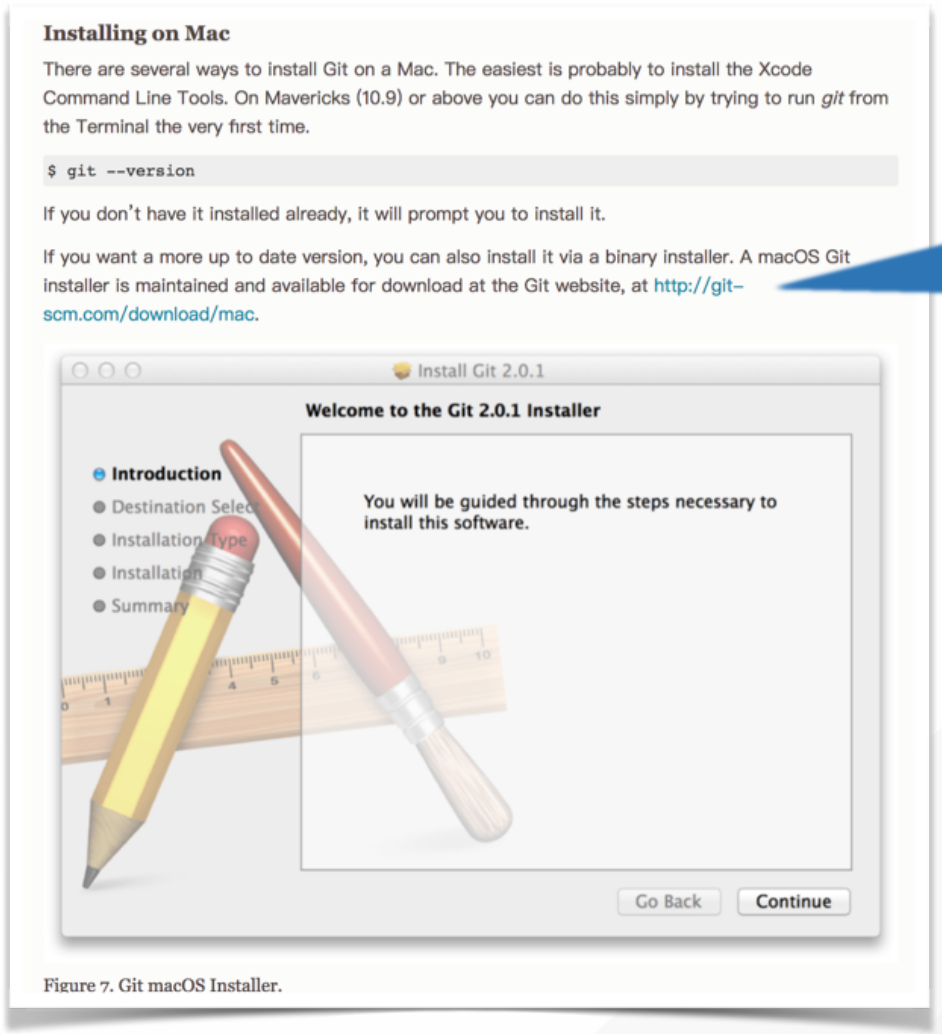
- Git 安装
- 初次运行 Git 前的最小配置
- Git 工作流程
- Git 基本命令
- Git 使用技巧

2.1 Git 安装



https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

2.1 Git 安装——在 Mac 上安装 Git



Installing on Mac

There are several ways to install Git on a Mac. The easiest is probably to install the Xcode Command Line Tools. On Mavericks (10.9) or above you can do this simply by trying to run *git* from the Terminal the very first time.

```
$ git --version
```

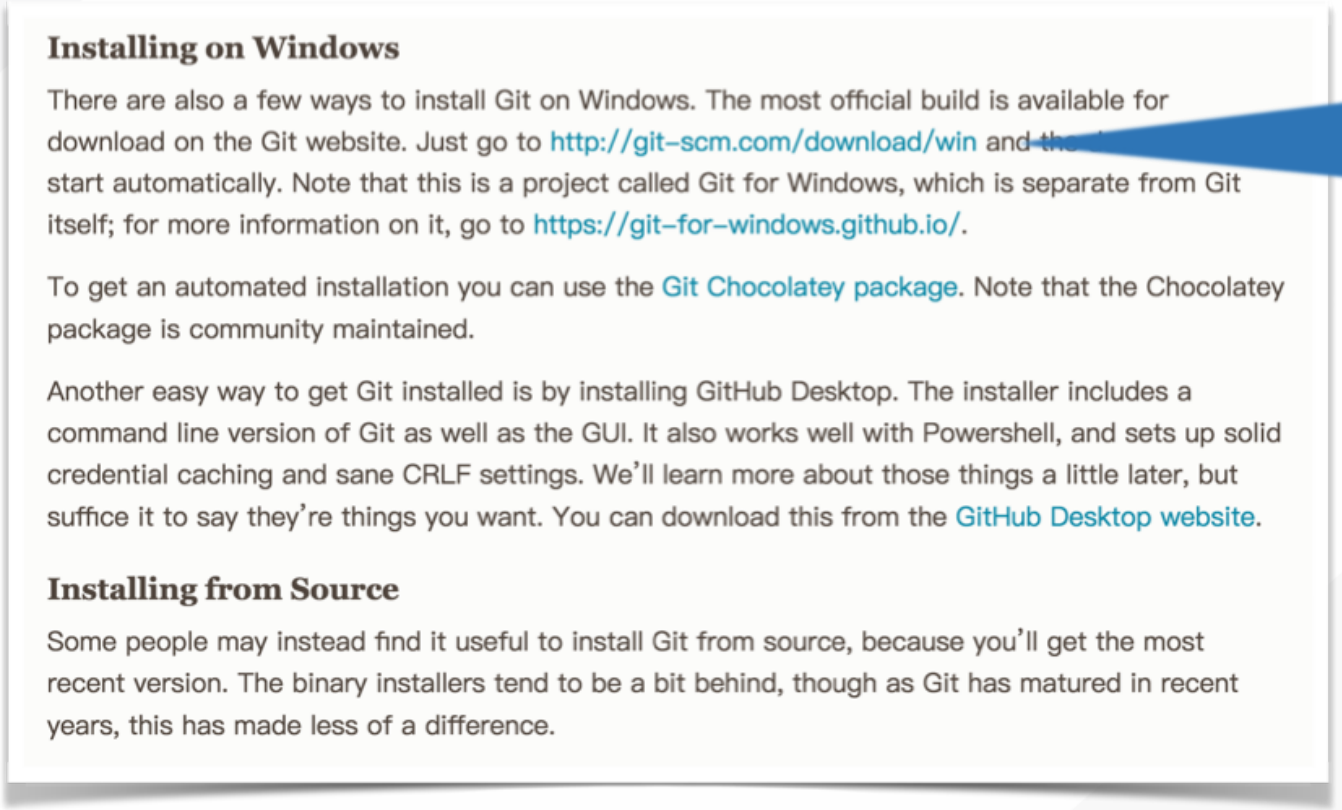
If you don't have it installed already, it will prompt you to install it.

If you want a more up to date version, you can also install it via a binary installer. A macOS Git installer is maintained and available for download at the Git website, at http://git-scm.com/download/mac.

Figure 7. Git macOS Installer.

我们选择二进制安装方式，下载dmg文件并打开安装，按默认方式一路回车即可。

2.1 Git 安装——在 Windows 上安装 Git



Installing on Windows

There are also a few ways to install Git on Windows. The most official build is available for download on the Git website. Just go to http://git-scm.com/download/win and the download will start automatically. Note that this is a project called Git for Windows, which is separate from Git itself; for more information on it, go to https://git-for-windows.github.io/.

To get an automated installation you can use the Git Chocolatey package. Note that the Chocolatey package is community maintained.

Another easy way to get Git installed is by installing GitHub Desktop. The installer includes a command line version of Git as well as the GUI. It also works well with Powershell, and sets up solid credential caching and sane CRLF settings. We'll learn more about those things a little later, but suffice it to say they're things you want. You can download this from the GitHub Desktop website.

Installing from Source

Some people may instead find it useful to install Git from source, because you'll get the most recent version. The binary installers tend to be a bit behind, though as Git has matured in recent years, this has made less of a difference.

点击链接，自动帮你下载最新的安装包。

网速慢的同学请移步到国内镜像：https://pan.baidu.com/share/init?surl=EoqDL7lB3V1NX7x_5vYlUQ 提取码: 7w3y

2.1 Git 安装——检查安装结果

在 bash 下执行下面的命令，看是否返回 Git 的版本。

```
$ git --version
```

2.2 初次运行 Git 前的最小配置

当安装完 Git 应该做的第一件事就是设置你的用户名与邮箱！

```
$ git config --global user.name "Your Name"
$ git config --global user.email  "Your Email"
```

config 的三个作用域

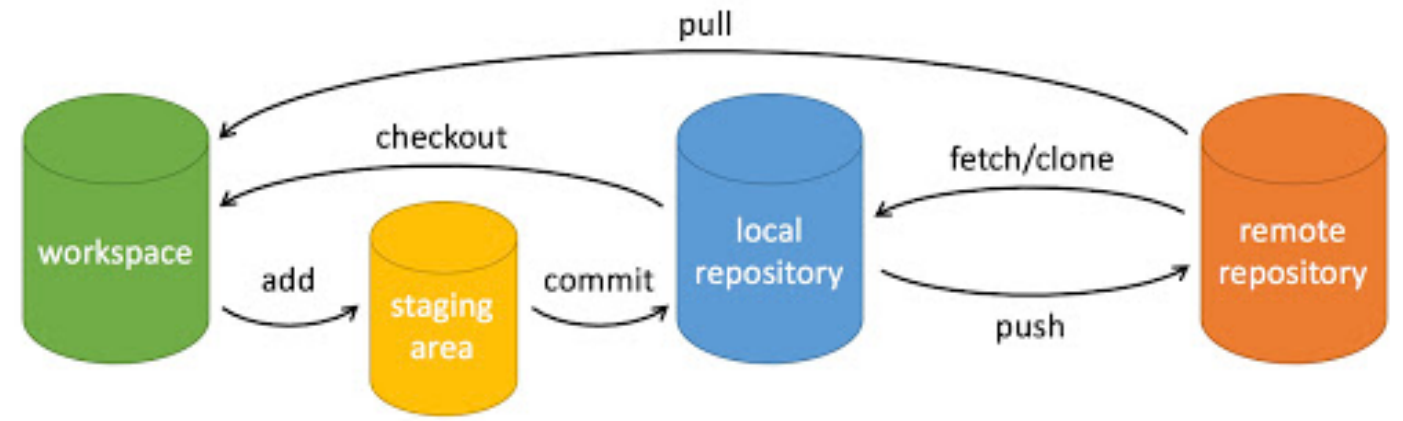
缺省等同于 local，优先级：local > global > system

$ git config --local （local 只对仓库有效）
$ git config --global（global 对登录用户所有仓库有效）
$ git config --system（system 对系统的所有用户有效）

显示 config 的配置，加 --list

$ git config --list --local
$ git config --list --global
$ git config --list --system

2.3 Git 工作流程



- workspace：工作区
- staging area：暂存区
- local repository：本地仓库
- remote repository：远程仓库

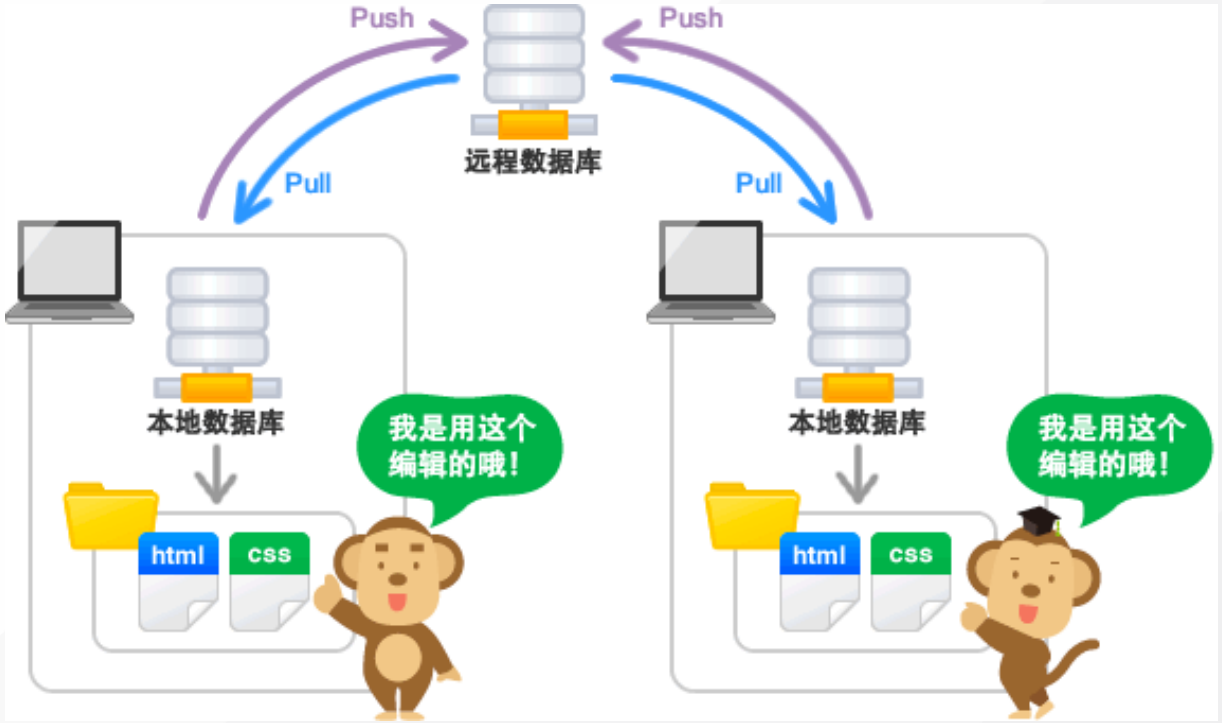
2.4 Git 基本命令——什么是仓库？

仓库又名版本库（或数据库），你可以简单理解成一个目录，这个目录里面的所有文件都可以被 Git 管理（.git 隐藏文件夹）起来，每个文件的修改、删除操作，Git 都能跟踪，以便任何时刻都可以追踪历史，或者在将来某个时刻可以“还原”。

Git 仓库分为远程仓库和本地仓库两种：

远程仓库：配有专用的服务器，为了多人共享而建立的仓库。

本地仓库：为了方便个人使用，在自己的电脑上配置的仓库。



2.4 Git 基本命令——创建 Git 本地仓库

两种方式：

1. 用 Git 之前还没有项目代码

```
$ cd 某个文件夹名
$ git init 项目文件夹名
$ cd 项目文件夹名
```

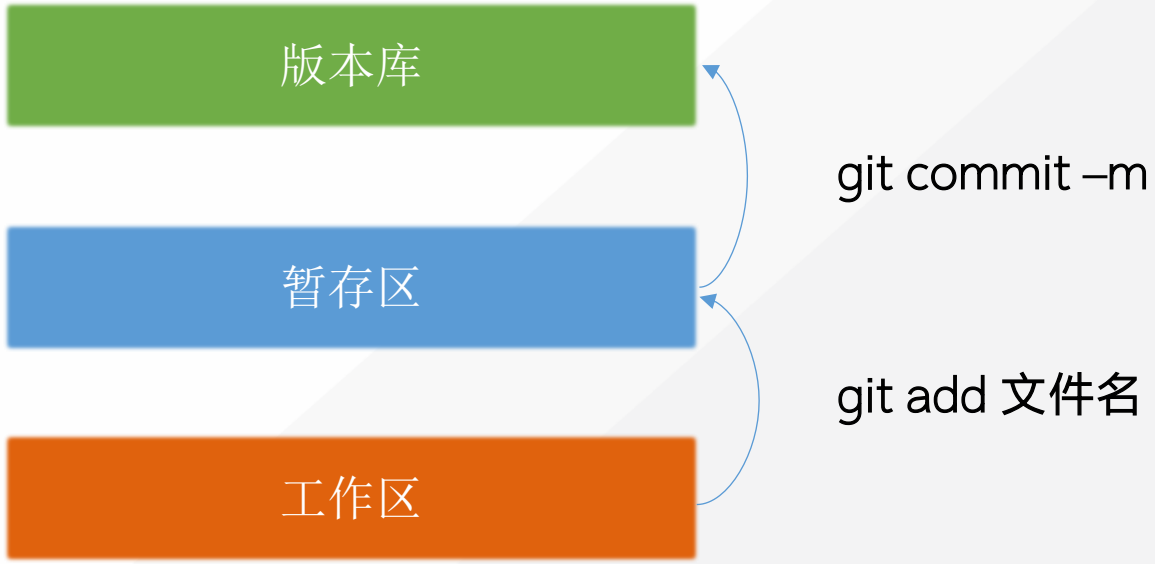
2. 用 Git 之前已经有项目代码

```
$ cd 项目代码所在的文件夹
$ git init
```

2.4 Git 基本命令——往本地仓库里添加文件

4次提交，一个简单的静态网页生成了。

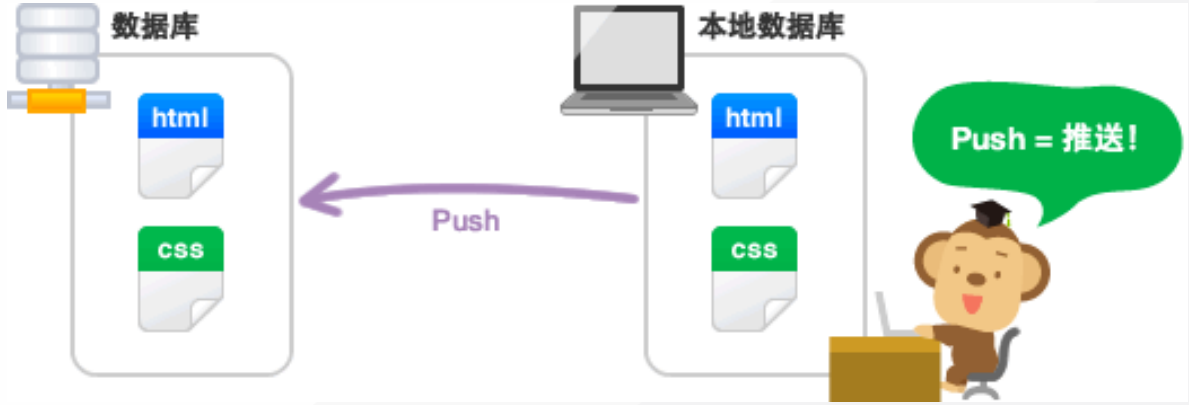
1. 加入 index.html
2. 加入 style.css
3. 加入 script.js
4. 修改 index.html 和 style.css



2.4 Git 基本命令——推送到远程仓库

创建远程仓库——https://github.com/

推送（push）：



1. 使用 remote 指令添加到远程仓库

$ git remote add origin 远程仓库URL

2. 使用 push 指令把本地仓库的所有内容推送到远程仓库

$ git push –u origin master

PS：当被要求输入用户名和密码，请使用你在 Github 上的用户名和密码。

2.4 Git 基本命令——从远程仓库克隆

前面都是先有本地仓库，后有远程仓库然后如何关联远程仓库的。现在，假设我们从零开发，那么最好的方式是先创建远程仓库，然后，从远程仓库克隆。

克隆（clone）：把远程仓库克隆到本地。

```
$ git clone 远程仓库URL
```

查看远程仓库：

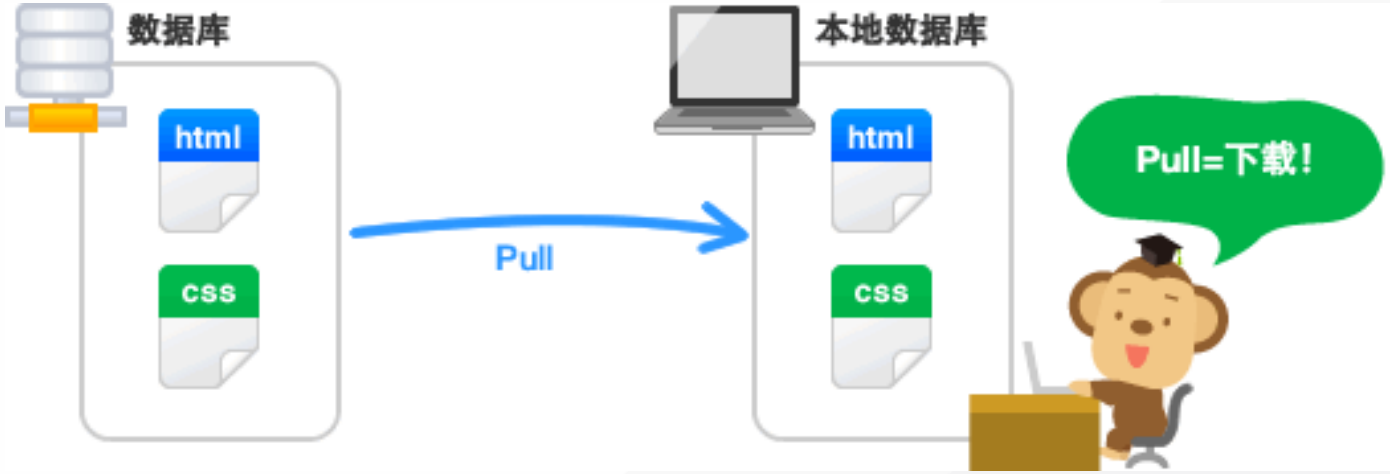
```
$ git remote -v
```

2.4 Git 基本命令——从远程仓库拉取

若是共享的远程仓库由多人同时作业，那么作业完毕后所有人都要把修改推送到远程仓库。然后，自己的本地仓库也需要更新其他人推送的变更内容。

拉取（pull）：把远程仓库内容更新到本地仓库。

$ git pull



2.5 Git 使用技巧——给文件重命名的简便方法

常规做法：

```
$ git clone 远程仓库地址
$ cd 远程仓库目录
$ touch README.txt
$ git add README.txt
$ git commit -m "add readme.txt"
$ git push
$ git log
```

简便做法：

```
$ git mv README.txt README.md
$ git status
$ git commit -m "rename readme"
$ git log
```

2.5 Git 使用技巧——正确删除文件的方法

常规做法：

```
$ rm readme.md
$ git add readme.md
```

简便做法：

```
$ git rm readme.md
```

2.5 Git 使用技巧——如何指定不需要 Git 管理的文件？

通常，有的文件比如日志，临时文件，编译的中间文件等不需要提交到代码仓库，这时就要设置相应的忽略规则，来忽略这些文件的提交。

.gitignore 文件的作用就是告诉 Git，push 的时候忽略指定的文件夹或者文件，在该文件中每一行指定一个忽略规则。例如：

*.log：忽略所有 .log 文件
**/foo：忽略 /foo, a/foo, a/b/foo 等（ **匹配多级目录）
a/**/b：忽略 a/b, a/x/b, a/x/y/b 等
.DS_Store：忽略 .DS_Store 文件
bin/：忽略 bin 目录以及该目录下的所有内容，但是不忽略 bin 文件
/bin：忽略根目录下的 bin 文件

三. GitHub 基础

- GitHub 为什么这么火？
- GitHub 有哪些核心功能？
- 注册一个 GitHub 账号
- 在 GitHub 上创建个人仓库
- 把本地仓库同步到 GitHub