

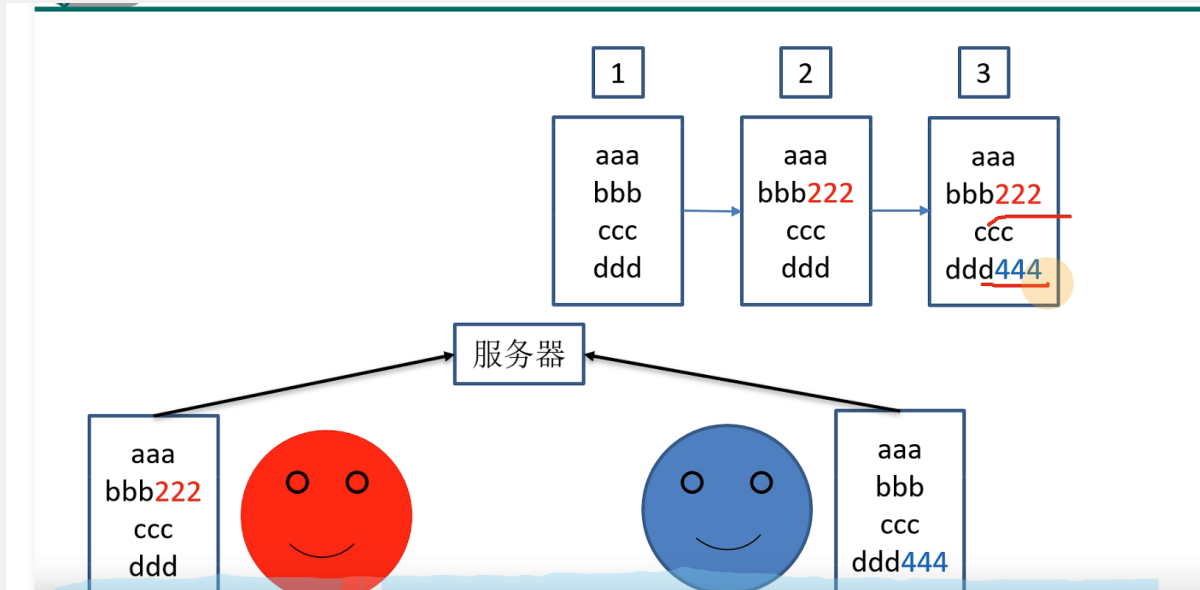
Git

git概述

- git是一个免费开源的**分布式版本控制系统**

版本控制?

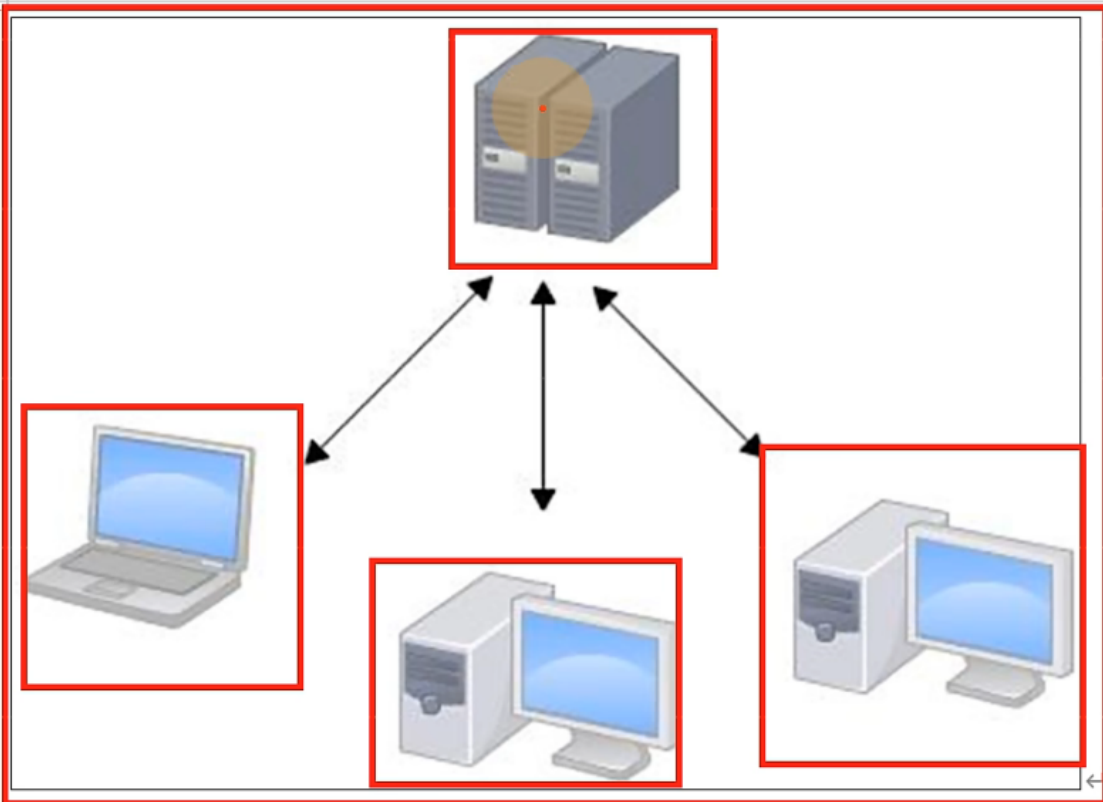
- 能够保存和切换文件的历史版本
- 防止后人修改覆盖前人修改,由版本控制系统实现不同版本文件的合并



分布式版本控制or集中式版本控制?

集中式版本控制:

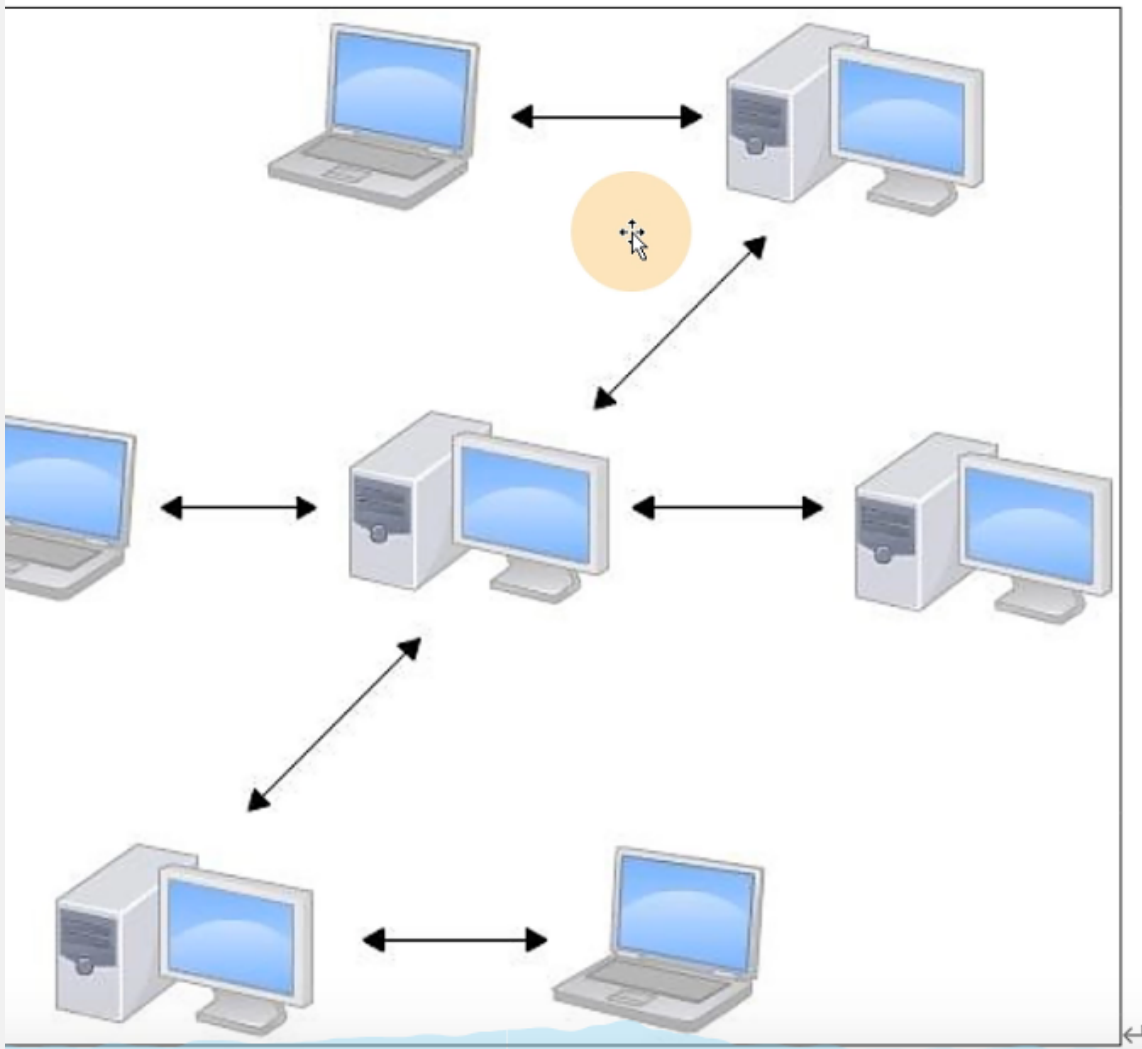
所有人从中央服务器上下载文件后,将不同版本的文件提交到中央服务器;



优点: 便于管理员分配权限

缺点: 中心服务器的宕机导致项目无法更新

分布式版本控制:

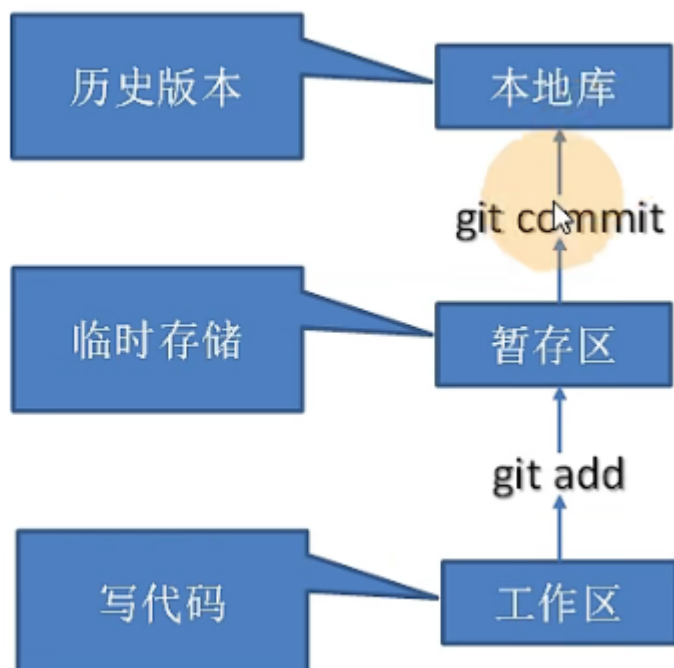


每个人的PC机就作为一个版本控制系统, 版本控制是在本地进行的, 每个客户端保存的是完整的项目

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



工作区和缓存区的代码可以删除,但是**提交到本地库的历史版本无法删除**

代码托管中心

1.6 Git 和代码托管中心

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（外网）

✓ Gitee 码云（国内网站）

git常用命令

1. 用户签名：用于识别不同用户（**必须要设置，否则提交代码会报错**）

<code>git config --global user.name 用户名</code>	设置用户签名
<code>git config --global user.email 邮箱</code>	设置用户签名

2. 本地库初始化：

```
xyx@LAPTOP-KA17J01D MINGW64 /d/gitSpace/git_demo
$ git init
```

3. 查看本地库状态：

```

xyx@LAPTOP-KA17J01D MINGW64 /d/gitSpace/git_demo (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

```

第一行表示所在分支；第二行表示当前库为空；第三行表示没有可追踪的文件

```

xyx@LAPTOP-KA17J01D MINGW64 /d/gitSpace/git_demo (master)
$ vim hello.txt

xyx@LAPTOP-KA17J01D MINGW64 /d/gitSpace/git_demo (master)
$ ll
total 1
-rw-r--r-- 1 xyx 197609 147 Nov 13 20:34 hello.txt

xyx@LAPTOP-KA17J01D MINGW64 /d/gitSpace/git_demo (master)
$ cat hello.txt
hello xyx!hello git!
hello xyx!hello git!
hello xyx!hello git!
hello xyx!hello git!
hello xyx!hello git!
hello xyx!hello git!
hello xyx!hello git!

```

在当前demo创建文件

```

xyx@LAPTOP-KA17J01D MINGW64 /d/gitSpace/git_demo (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    hello.txt

nothing added to commit but untracked files present (use "git add" to track)

```

显示可追踪文件hello.txt,此时暂存区还没有保存文件

```

xyx@LAPTOP-KA17J01D MINGW64 /d/gitSpace/git_demo (master)
$ git add hello.txt
warning: in the working copy of 'hello.txt', LF will be replaced by CRLF the next time Git touches it

xyx@LAPTOP-KA17J01D MINGW64 /d/gitSpace/git_demo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   hello.txt

```

调用"git add"命令，将工作区的文件保存到暂存区，此时暂存区显示创建了新文件

```

xyx@LAPTOP-KA17J01D MINGW64 /d/gitSpace/git_demo (master)
$ git rm --cached hello.txt
rm 'hello.txt'

xyx@LAPTOP-KA17J01D MINGW64 /d/gitSpace/git_demo (master)
$ ll
total 1
-rw-r--r-- 1 xyx 197609 147 Nov 13 20:34 hello.txt

```

调用"git rm --cached hello.txt"删除暂存区文件，工作区的文件不会被删除

4. 提交本地库： git commit -m "版本备注" [文件名]

```

xyx@LAPTOP-KA17J01D MINGW64 /d/gitSpace/git_demo (master)
$ git commit -m "first commit" hello.txt
warning: in the working copy of 'hello.txt', LF will be rep
t time Git touches it
[master (root-commit) 51fa2cd] first commit
1 file changed, 7 insertions(+)
create mode 100644 hello.txt

xyx@LAPTOP-KA17J01D MINGW64 /d/gitSpace/git_demo (master)
$ git status
On branch master
nothing to commit, working tree clean

```

显示没有待提交文件，同时工作区也被清空

5. 查看本地库历史版本状态： git reflog

```

nothing to commit, working tree clean

asus@LAPTOP-Layne MINGW64 /d/Git-Space/git-demo (master)
$ git reflog
5770506 (HEAD -> master) HEAD@{0}: commit: second commit
965c6a1 HEAD@{1}: commit (initial): first commit

```

当前指针指向第三个版本

6. 将指针指向不同的历史版本： git reset --hard [版本号]

```

xyx@LAPTOP-KA17J01D MINGW64 /d/gitSpace/g
$ git reset --hard 3731b4a
HEAD is now at 3731b4a second commit

```

7. 分支的创建、切换、合并

4.3 分支的操作

命令名称	作用
git branch 分支名	创建分支
git branch -v	查看分支
git checkout 分支名	切换分支
git merge 分支名	把指定的分支合并到当前分支上

gitHub 基本操作

1. 创建远程库别名：

git remote -v 查看当前所有远程地址别名

git remote add 别名 远程地址

2) 案例实操



```
Layne@LAPTOP-Layne MINGW64 /d/Git-Space/SH0720 (master)
$ git remote -v
Layne@LAPTOP-Layne MINGW64 /d/Git-Space/SH0720 (master)
$ git remote add ori https://github.com/atguigu Yueyue/git-shTest.git
Layne@LAPTOP-Layne MINGW64 /d/Git-Space/SH0720 (master)
$ git remote -v
ori      https://github.com/atguigu Yueyue/git-shTest.git (fetch)
ori      https://github.com/atguigu Yueyue/git-shTest.git (push)
```

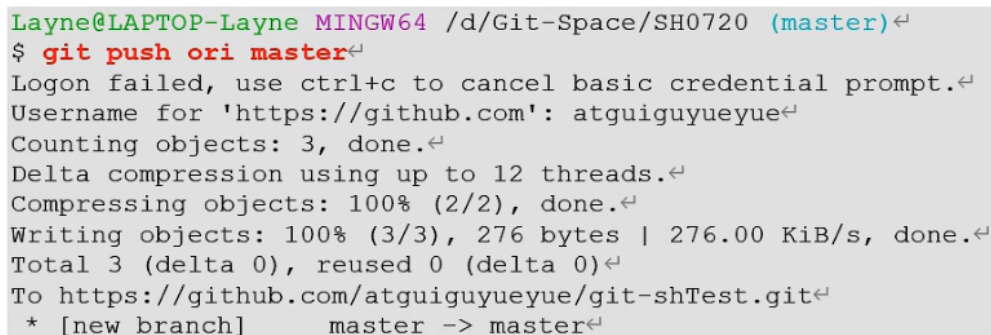
2. 推送本地库至远程库：

▪ 6.2.2 推送本地分支到远程仓库

1) 基本语法

git push 别名 分支

2) 案例实操



```
Layne@LAPTOP-Layne MINGW64 /d/Git-Space/SH0720 (master)
$ git push ori master
Logon failed, use ctrl+c to cancel basic credential prompt.
Username for 'https://github.com': atguigu Yueyue
Counting objects: 3, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 276 bytes | 276.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/atguigu Yueyue/git-shTest.git
* [new branch] master -> master
```

3. 拉取远程库到本地库：只需要将推送中的“push” 更改为“pull” 拉取和推送都需要制定一个分支，拉取指定拉取远程库的哪个分支，推送指定推送本地库的哪个分支

4. 克隆fork远程库到本地库：克隆是复制远程库的所有内容，所以无须指定分支。