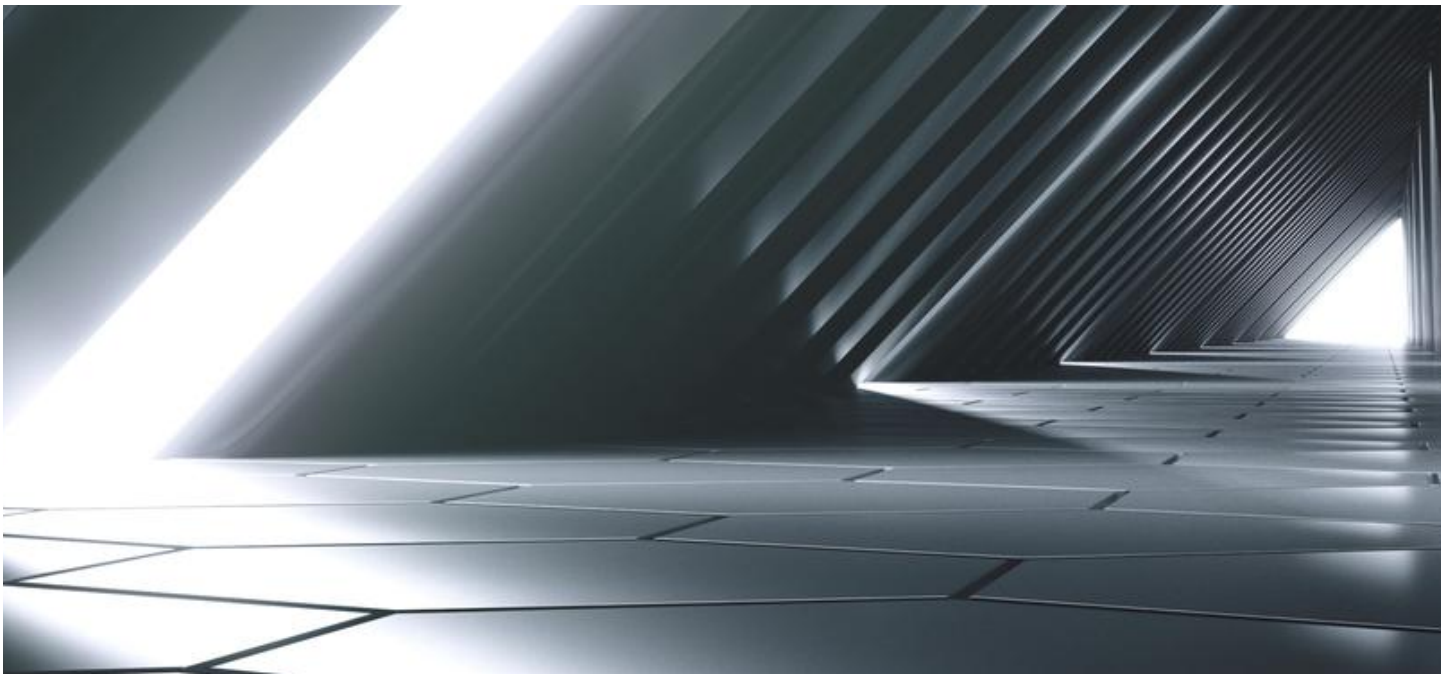


Git

Git 概述

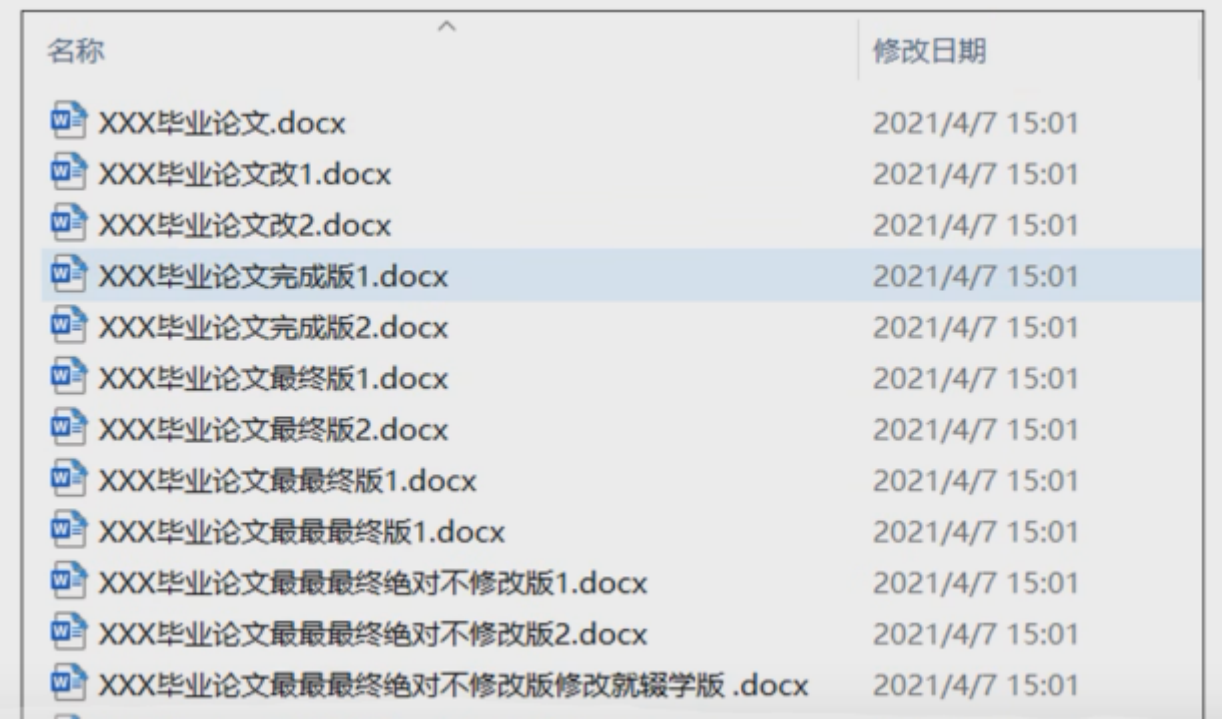
Git 是一个免费的、开源的分布式版本控制系统，可以快速高效地处理从小型到大型的各种项目。

Git 易于学习，占用空间小，性能极快。它具有低成本的本地库，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。

版本控制介绍

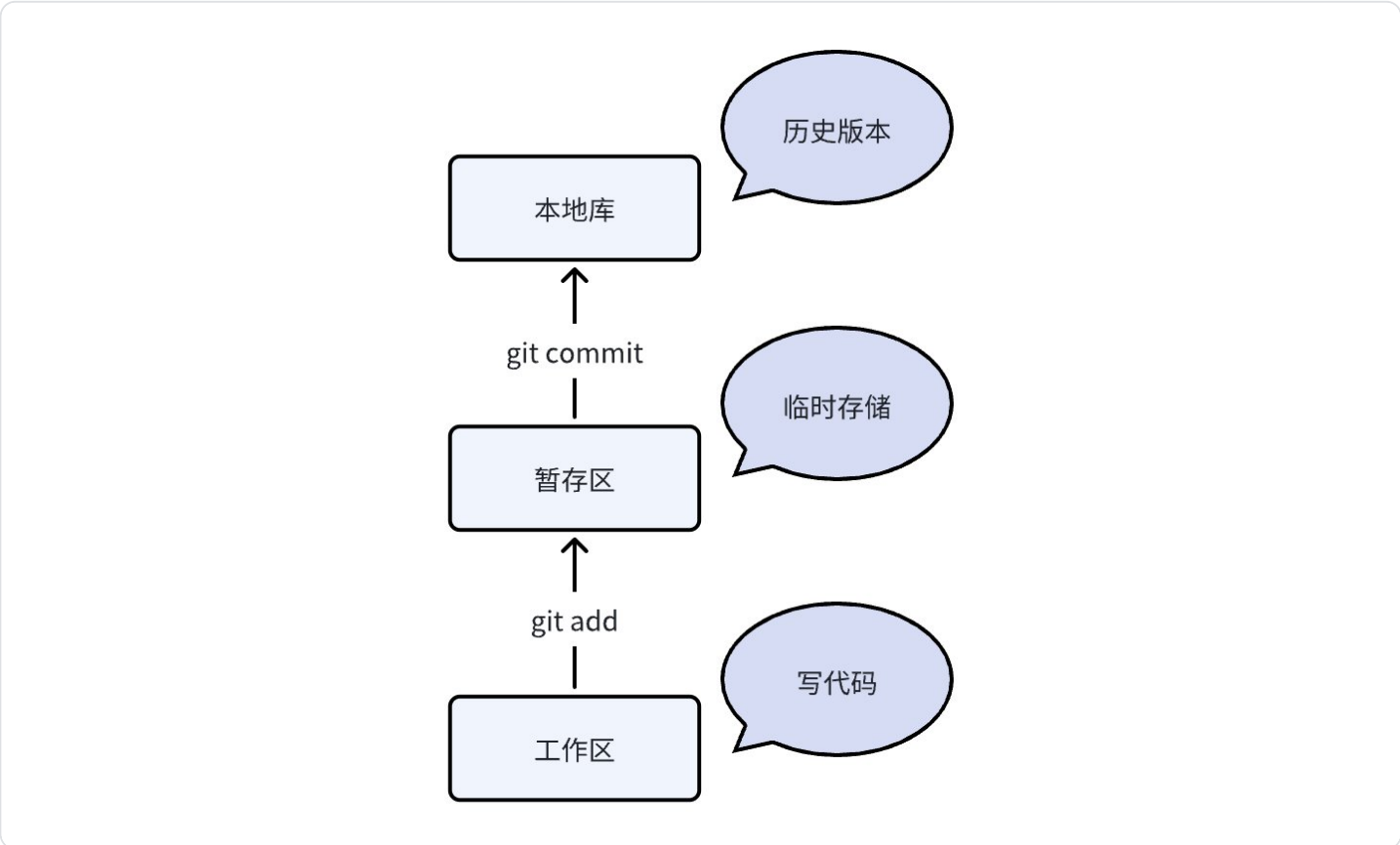
版本控制是一种记录文件内容变化，以便将来查阅特定版本修订情况的系统。

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。



名称	修改日期
XXX毕业论文.docx	2021/4/7 15:01
XXX毕业论文改1.docx	2021/4/7 15:01
XXX毕业论文改2.docx	2021/4/7 15:01
XXX毕业论文完成版1.docx	2021/4/7 15:01
XXX毕业论文完成版2.docx	2021/4/7 15:01
XXX毕业论文最终版1.docx	2021/4/7 15:01
XXX毕业论文最终版2.docx	2021/4/7 15:01
XXX毕业论文最最终版1.docx	2021/4/7 15:01
XXX毕业论文最最最终版1.docx	2021/4/7 15:01
XXX毕业论文最最最终绝对不修改版1.docx	2021/4/7 15:01
XXX毕业论文最最最终绝对不修改版2.docx	2021/4/7 15:01
XXX毕业论文最最最终绝对不修改版修改就辍学版 .docx	2021/4/7 15:01

> **为什么需要版本控制：个人开发过渡到团队协作开发。**



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。

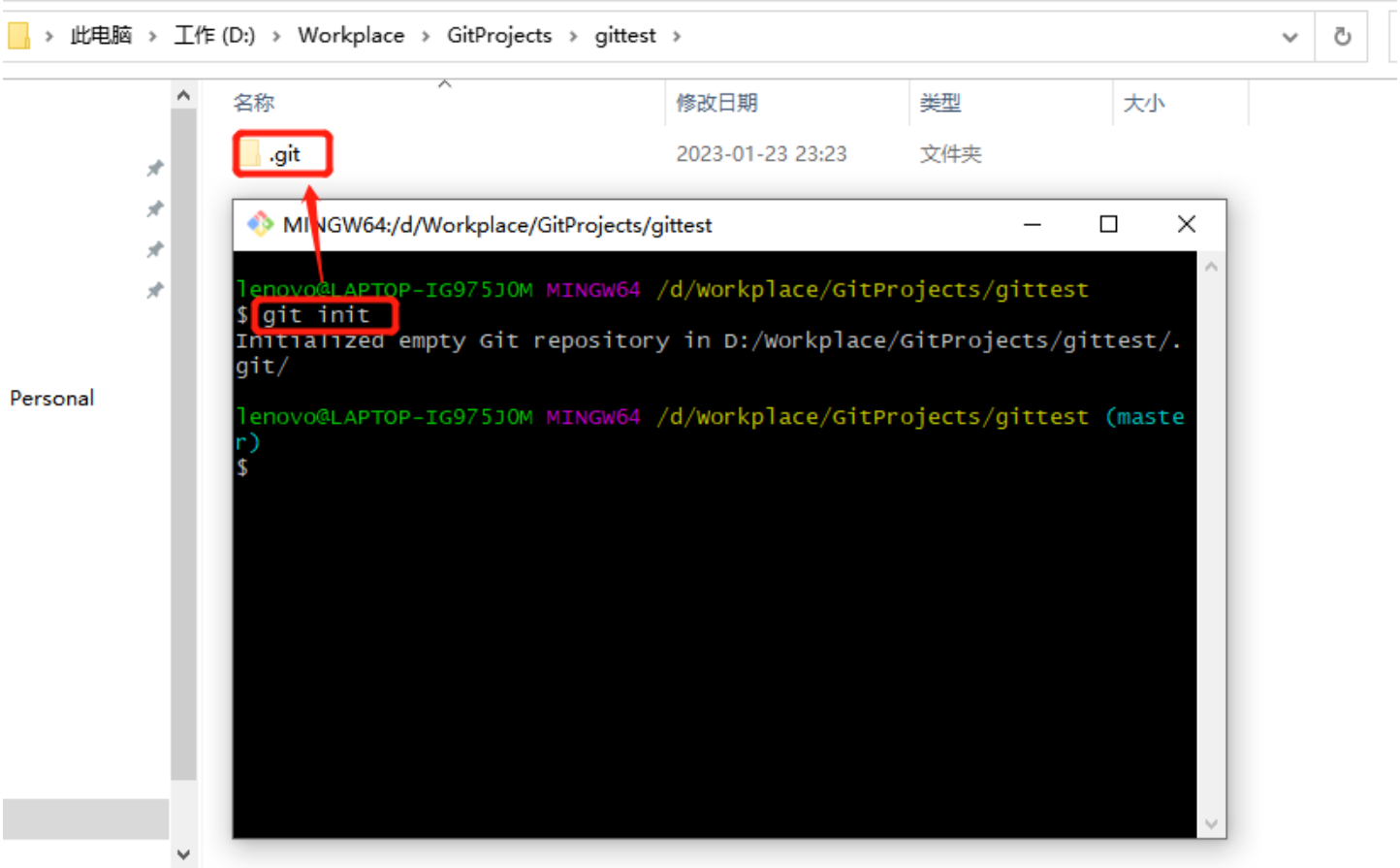
- 局域网
  - GitLab

- 互联网
  - GitHub（外网）
  - Gitee 码云（国内网站）

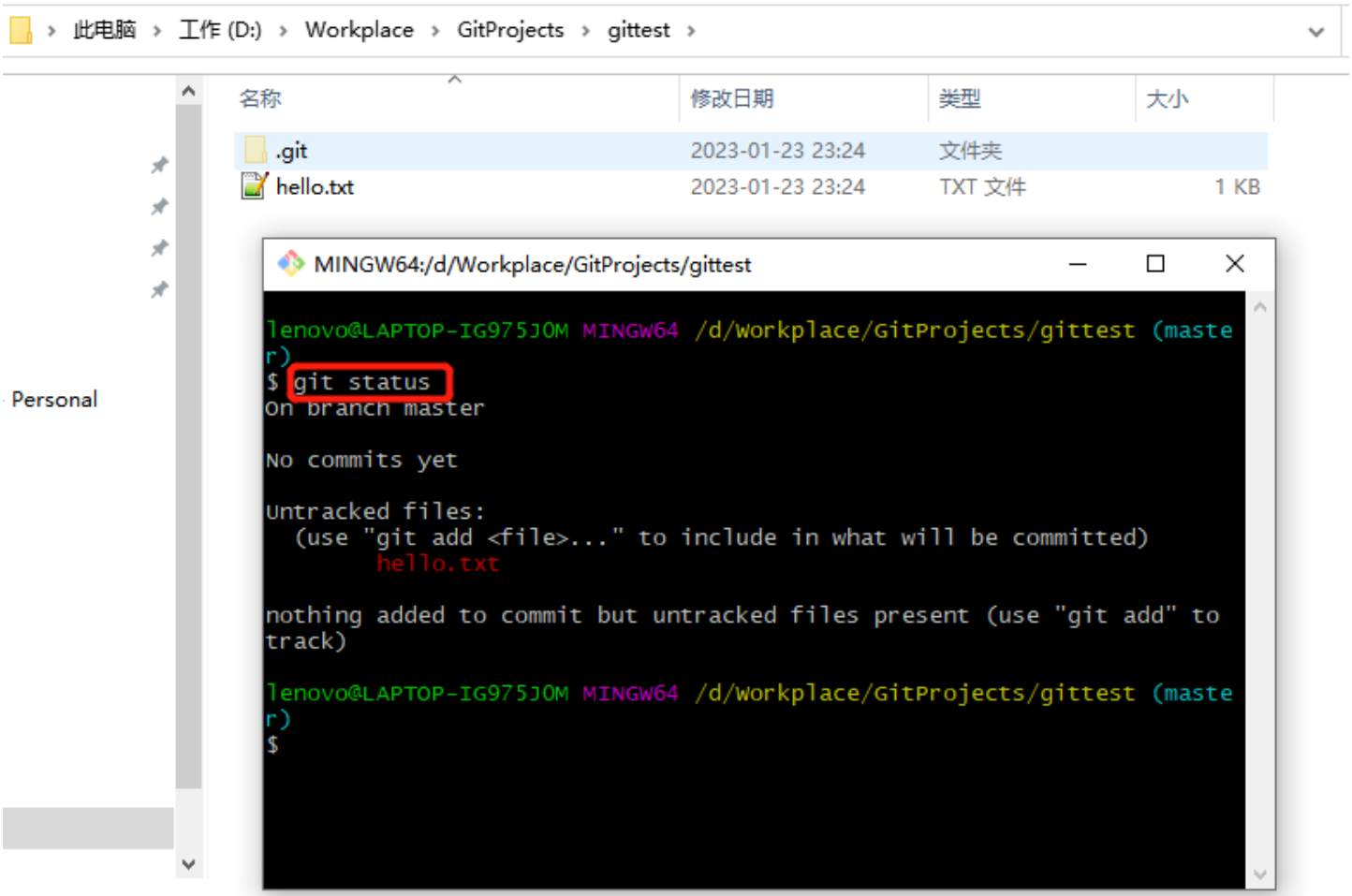
Git 常用命令

命令名称	作用
`git config --global user.name 用户名`	设置用户签名
`git config --global user.email 邮箱`	设置用户邮箱
`git init`	初始化本地库
`git status`	查看本地库状态
`git add 文件名`	添加到暂存区
`git commit -m "日志信息" 文件名`	提交到本地库
`git reflog`	查看历史记录
`git reset --hard 版本号`	版本穿梭

初始化 Git


此电脑 › 工作 (D:) › Workplace › GitProjects › gittest ›
名称
修改日期
类型
大小
.git
2023-01-23 23:23
文件夹
Personal
MINGW64:/d/Workplace/GitProjects/gittest
lenovo@LAPTOP-IG975JOM MINGW64 /d/workplace/GitProjects/gittest
$ git init
Initialized empty Git repository in D:/Workplace/GitProjects/gittest/.
git/
lenovo@LAPTOP-IG975JOM MINGW64 /d/workplace/GitProjects/gittest (maste
r)
$

查看 Git 状态


此电脑 › 工作 (D:) › Workplace › GitProjects › gittest ›
名称
修改日期
类型
大小
.git
2023-01-23 23:24
文件夹
hello.txt
2023-01-23 23:24
TXT 文件
1 KB
Personal
MINGW64:/d/Workplace/GitProjects/gittest
lenovo@LAPTOP-IG975JOM MINGW64 /d/workplace/GitProjects/gittest (maste
r)
$ 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)
lenovo@LAPTOP-IG975JOM MINGW64 /d/workplace/GitProjects/gittest (maste
r)
$

将文件从工作区添加到暂存区

```
MINGW64:/d/Workplace/GitProjects/gittest

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (maste
r)
$ git add hello.txt
warning: LF will be replaced by CRLF in hello.txt.
The file will have its original line endings in your working directory

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (maste
r)
$ git status
On branch master

No commits yet

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


lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (maste
r)
$
```

> 通过 `git rm --cached hello.txt` 可以将“hello.txt”文件从暂存区中删除。

提交到本地库

```
MINGW64:/d/Workplace/GitProjects/gittest

No commits yet

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


lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (maste
r)
$ git commit -m "V1" hello.txt
warning: LF will be replaced by CRLF in hello.txt.
The file will have its original line endings in your working directory
[master (root-commit) caab8fa] V1
 1 file changed, 1 insertion(+)
 create mode 100644 hello.txt

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (maste
r)
$ git status
On branch master
nothing to commit, working tree clean

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (maste
r)
$
```

查看操作日志

```
MINGW64:/d/Workplace/GitProjects/gittest

lenovo@LAPTOP-IG975J0M MINGW64 /d/workplace/GitProjects/gittest (master)
$ git reflog
caab8fa (HEAD -> master) HEAD@{0}: commit (initial): V1

lenovo@LAPTOP-IG975J0M MINGW64 /d/workplace/GitProjects/gittest (master)
$ git log
commit caab8fa9e1aac05eede0787bf685d4eebdad13a6 (HEAD -> master)
Author: wujiaxuan0729 <wujiaxuan_will@163.com>
Date:   Mon Jan 23 23:28:38 2023 +0800

    V1

lenovo@LAPTOP-IG975J0M MINGW64 /d/workplace/GitProjects/gittest (master)
$
```

简要模式

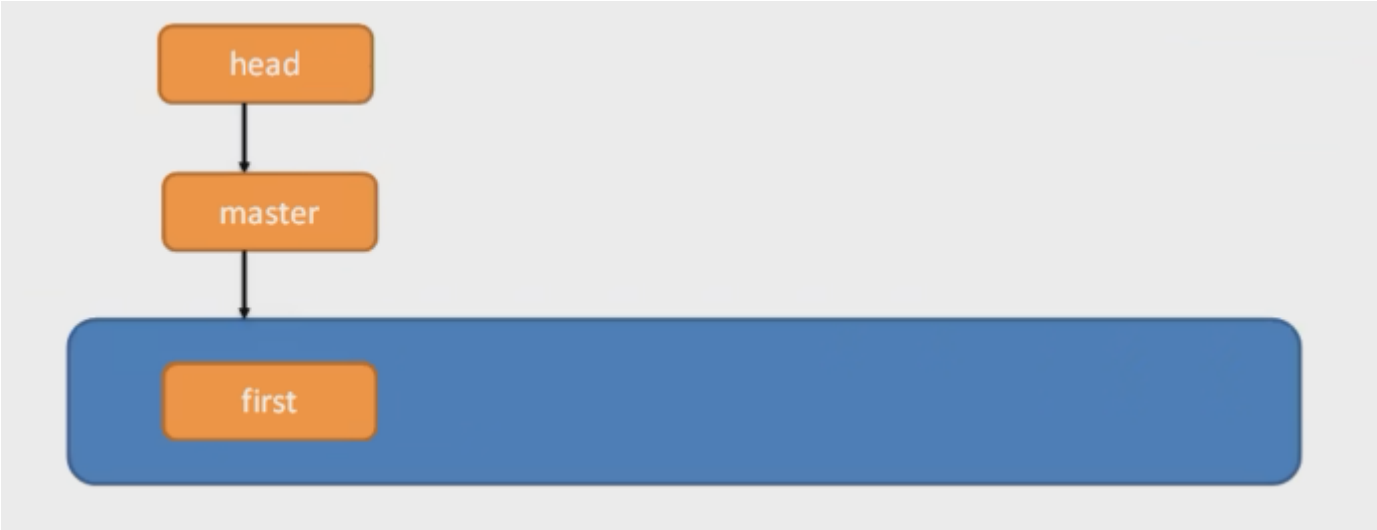
详细模式

文件修改

```
MINGW64:/d/Workplace/GitProjects/gittest

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.txt

no changes added to commit (use "git add" and/or "git commit -a")

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (master)
$ git add hello.txt
warning: LF will be replaced by CRLF in hello.txt.
The file will have its original line endings in your working directory

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (master)
$ git commit -m "V2" hello.txt
warning: LF will be replaced by CRLF in hello.txt.
The file will have its original line endings in your working directory
[master 8ce1502] V2
 1 file changed, 1 insertion(+)

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (master)
$ git reflog
8ce1502 (HEAD -> master) HEAD@{0}: commit: V2
caab8fa HEAD@{1}: commit (initial): V1

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (master)
$ |
```

版本穿梭

```
MINGW64:/d/Workplace/GitProjects/gittest

lenovo@LAPTOP-IG975JOM MINGW64 /d/Workplace/GitProjects/gittest (master)
$ git reflog
8ce1502 (HEAD -> master) HEAD@{0}: commit: V2
caab8fa HEAD@{1}: commit (initial): V1

lenovo@LAPTOP-IG975JOM MINGW64 /d/Workplace/GitProjects/gittest (master)
$ git reset --hard caab8fa
HEAD is now at caab8fa V1

lenovo@LAPTOP-IG975JOM MINGW64 /d/Workplace/GitProjects/gittest (master)
$ git reflog
caab8fa (HEAD -> master) HEAD@{0}: reset: moving to caab8fa
8ce1502 HEAD@{1}: commit: V2
caab8fa (HEAD -> master) HEAD@{2}: commit (initial): V1
```



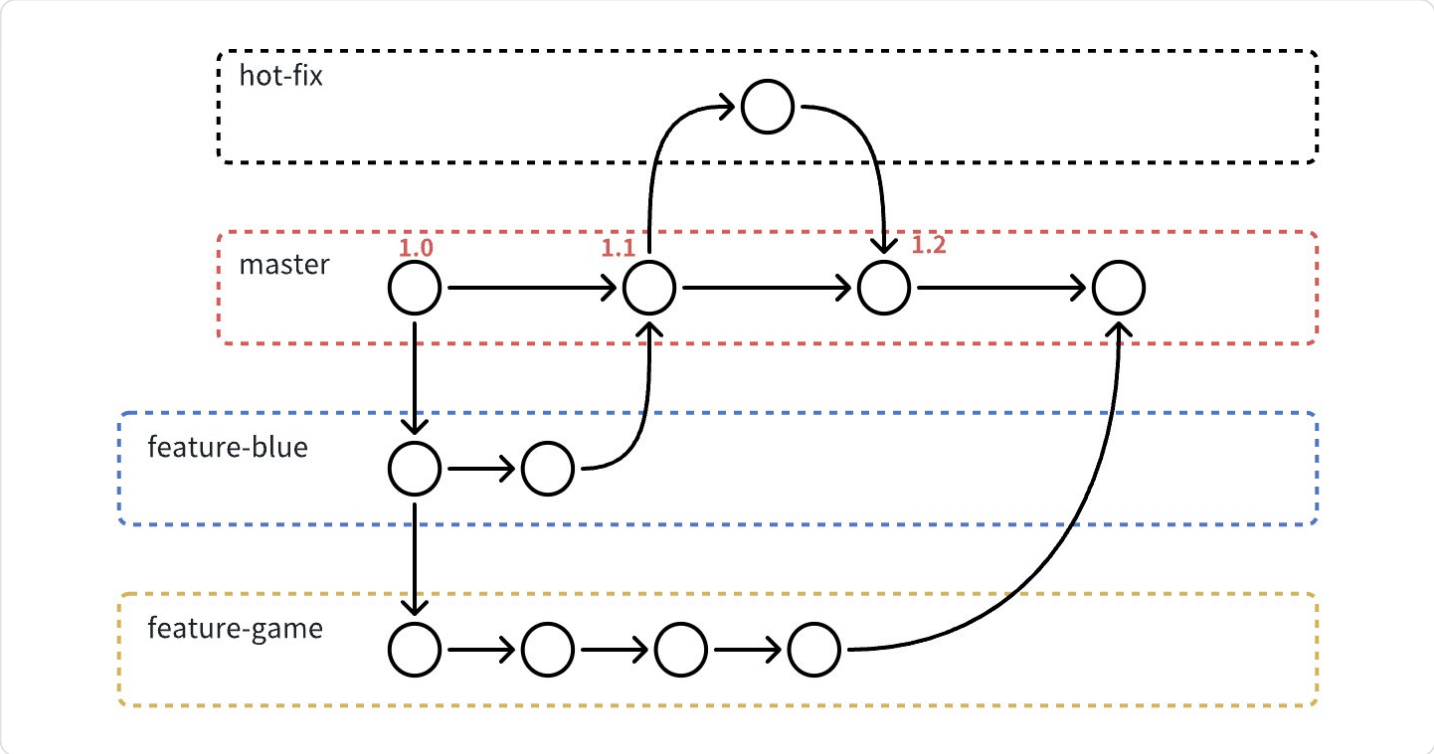
> **first**是第一个版本，**master**代表**master**分支，**head**代表目前我们看到的样子。

Git 分支

什么是分支

在版本控制过程中，同时推进多个任务，为每个任务，我们就可以创建每个任务的单独分支。使用分支意味着程序员可以把自己的工作从开发主线上分离出来，开发自己分支的时候，不会影响主线分支的运行。

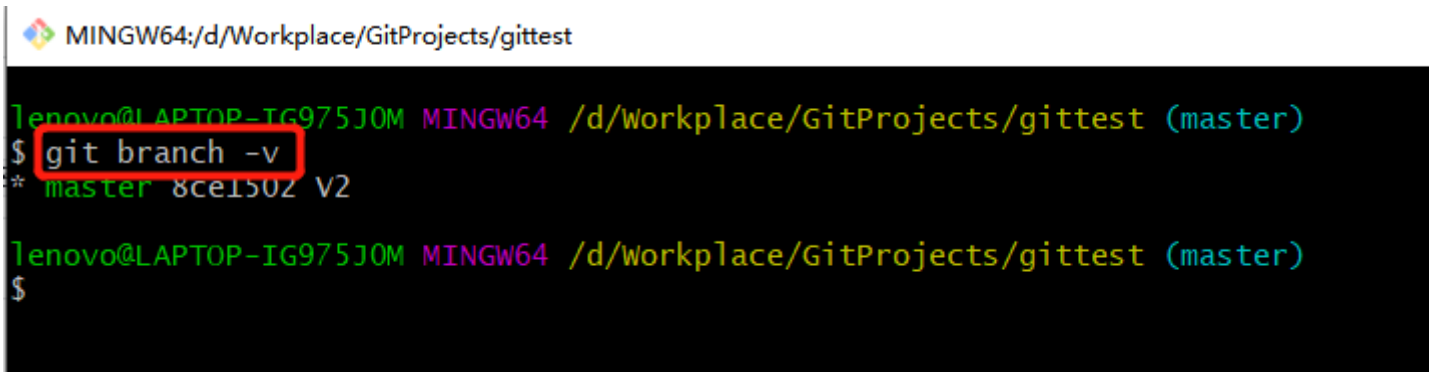
对于初学者而言，分支可以简单理解为副本，一个分支就是一个单独的副本。（分支底层其实也是指针的引用）



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

查看分支



创建分支

```
MINGW64:/d/Workplace/GitProjects/gittest

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (master)
$ git branch hot-fix

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (master)
$ git branch -v
  hot-fix 8ce1502 V2
* master  8ce1502 V2

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (master)
$
```

切换分支

```
MINGW64:/d/Workplace/GitProjects/gittest

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (master)
$ git checkout hot-fix
Switched to branch 'hot-fix'

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (hot-fix)
$
```

在 hot-fix 分支上修改 hello.txt

```
MINGW64:/d/Workplace/GitProjects/gittest

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (hot-fix)
$ git status
On branch hot-fix
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.txt

no changes added to commit (use "git add" and/or "git commit -a")

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (hot-fix)
$ git add hello.txt

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (hot-fix)
$ git commit -m "hot-fix V1" hello.txt
[hot-fix a27c321] hot-fix V1
 1 file changed, 1 insertion(+)

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (hot-fix)
$ git status
On branch hot-fix
nothing to commit, working tree clean

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (hot-fix)
$
```

合并分支（无冲突）

```
MINGW64:/d/Workplace/GitProjects/gittest

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (hot-fix)
$ git checkout master
Switched to branch 'master'

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (master)
$ git merge hot-fix
Updating 8ce15b2..a27c321
Fast-forward
 hello.txt | 1 +
 1 file changed, 1 insertion(+)

lenovo@LAPTOP-IG975J0M MINGW64 /d/Workplace/GitProjects/gittest (master)
$ git status
On branch master
nothing to commit, working tree clean
```

合并分支（有冲突）

> ❗ **冲突产生的原因：**
>
> 合并分支时，如果两个分支在**同一个文件的同一个位置**有两套完全不同的修改。Git 无法替我们决定使用哪一个。必须**人为决定**新代码内容。

1. master 分支和 hot-fix 分支分别对同一个文件的同一个内容进行修改，并执行 `add` 和 `commit`

```
wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test (master)
$ cat paper.txt
第一天
写了1000字

第二天（hot-fix修改）
写了1000字

第三天（master修改，hot-fix也修改）
master写了500字
hot-fix写了600字

第四天（master修改）
master写了500字

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test (master)
$ git add .

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test (master)
$ git commit -m "第四天（master修改）"
[master c803631] 第四天（master修改）
 1 file changed, 3 insertions(+)
```

```
wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test (hot-fix)
$ cat paper.txt
第一天
写了1000字

第二天（hot-fix修改）
写了1000字

第三天（hot-fix修改）
写了600字

第四天（hot-fix修改）
写了1000字

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test (hot-fix)
$ git add .

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test (hot-fix)
$ git commit -m "第四天（hot-fix修改）"
[hot-fix 5bf250f] 第四天（hot-fix修改）
 1 file changed, 3 insertions(+)
```

2. 在master分支合并hot-fix分支，会报错。通过执行 `git status` 命令，可以看到**"both modified"**报错。

```
wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test (master)
$ git merge hot-fix
Auto-merging paper.txt
CONFLICT (content): Merge conflict in paper.txt
Automatic merge failed; fix conflicts and then commit the result.

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test (master|MERGING)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   paper.txt
```

3. 手动修改冲突文件，自行决定保留哪些内容。修改后重新执行 `add` 和 `commit` 命令。

```
wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test (master|MERGING)
$ cat paper.txt
第一天
写了1000字

第二天（hot-fix修改）
写了1000字

<<<<<<< HEAD
第三天（master修改，hot-fix也修改）
master写了500字
hot-fix写了600字

第四天（master修改）
master写了500字
=======
第三天（hot-fix修改）
写了600字

第四天（hot-fix修改）
写了1000字
>>>>>>> hot-fix

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test (master|MERGING)
$ vim paper.txt

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test (master|MERGING)
$ cat paper.txt
第一天
写了1000字

第二天（hot-fix修改）
写了1000字

第三天（master修改，hot-fix也修改）
master写了500字
hot-fix写了600字

第四天（master修改，hot-fix也修改）
master写了500字
hot-fix写了1000字

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test (master|MERGING)
$ git add .

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test (master|MERGING)
$ git commit -m "第四天，master和hot-fix都修改了"
[master 9e55f52] 第四天，master和hot-fix都修改了
```

4. 提交远程库

```
wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test (master)
$ git push origin master
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 32 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 923 bytes | 923.00 KiB/s, done.
Total 9 (delta 6), reused 0 (delta 0), pack-reused 0
remote: Powered by GITEE.COM [GNK-6.4]
To https://gitee.com/wujiaxuan0729/git_test.git
   d86dfac..9e55f52  master -> master
```

> origin 就是远程库的别名；master 代表当前的 master 分支。

知识点

master、hot-fix 其实都是指向具体版本记录的指针。而当前所在的分支，其实是由 HEAD 决定的。所以说，创建分支的本质就是多创建一个指针。

- HEAD 如果指向 master，那么我们现在就在 master 分支上；
- HEAD 如果指向 hot-fix，那么我们现在就在 hot-fix分支上。

所以说，切换分支的本质就是移动 HEAD 指针。

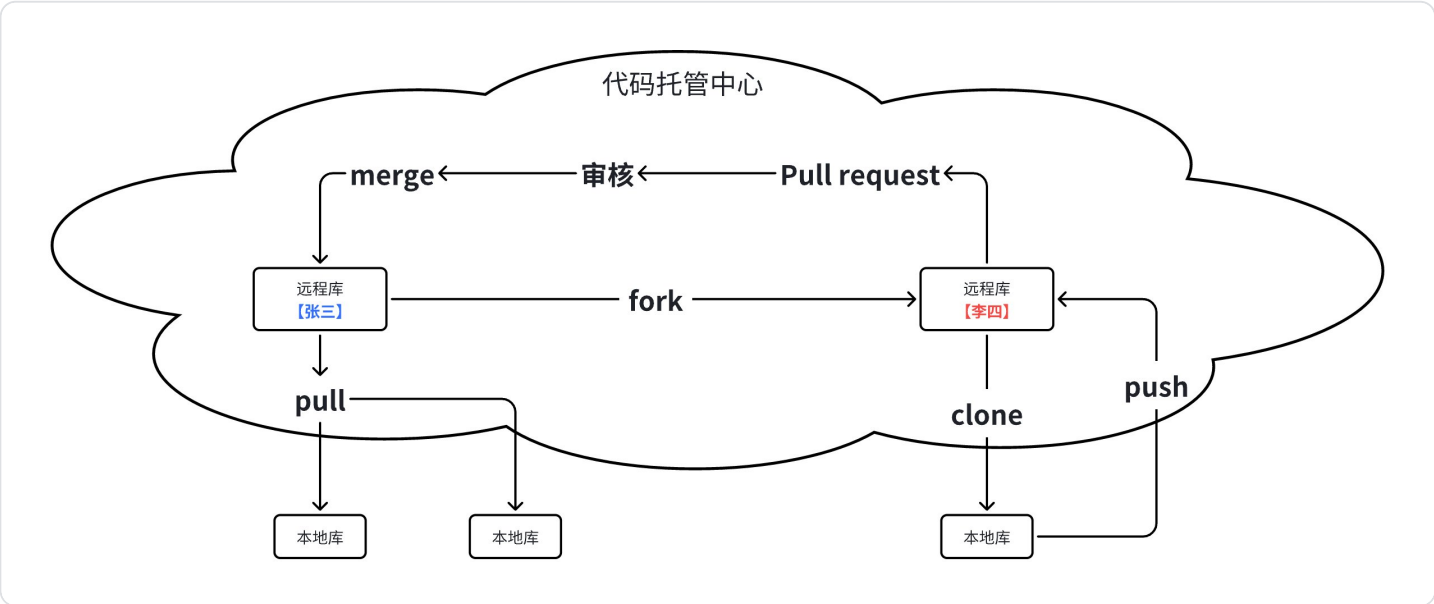
建立远程仓库

```
wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test2
$ git init
Initialized empty Git repository in C:/WJX/Workplace/GitProjects/git_test2/.git/

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test2 (master)
$ git remote add git-demo https://gitee.com/wujiaxuan0729/git_test_2.git

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test2 (master)
$ git remote -v
git-demo        https://gitee.com/wujiaxuan0729/git_test_2.git (fetch)
git-demo        https://gitee.com/wujiaxuan0729/git_test_2.git (push)

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test2 (master)
$ git pull git-demo master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (5/5), 351 bytes | 10.00 KiB/s, done.
From https://gitee.com/wujiaxuan0729/git_test_2
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> git-demo/master

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test2 (master)
$ touch wjx.txt

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test2 (master)
$ echo "wjx" >> wjx.txt

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test2 (master)
$ git add .
warning: in the working copy of 'wjx.txt', LF will be replaced by CRLF the next
time Git touches it

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test2 (master)
$ git commit -m "wjx"
[master f057024] wjx
 1 file changed, 1 insertion(+)
 create mode 100644 wjx.txt

wujia@WJX MINGW64 /c/WJX/Workplace/GitProjects/git_test2 (master)
$ git push git-demo master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Writing objects: 100% (3/3), 241 bytes | 241.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Powered by GITEE.COM [GNK-6.4]
To https://gitee.com/wujiaxuan0729/git_test_2.git
   328a46c..f057024  master -> master
```

> git-demo就是远程仓库的别名。master表示将本地的master分支推送到远程库中去。

跨团队协作

fork



> 1. fork后修改，提交。
> 2. 修改完后发起 pull-request。
> 3. 原作者审阅要是没问题的话，就可以合并修改。