9月17作业 - git进阶

一、主观题

1. git pull 和 git fetch的区别?

`git pull` —— **下载 + 自动合并**

作用：`git fetch` + `git merge` 的组合命令。它会先从远程拉取最新数据，然后**自动尝试合并到你当前所在的本地分支**。

可能直接修改你的工作区！在未保存或未提交本地修改时容易出问题。

`git fetch` —— **仅下载，不合并**

作用：从远程仓库下载最新的提交、分支、标签等对象到本地仓库，**但不会自动合并或修改你当前的工作区和分支**。

2. 两个历史不相同的仓库如何git pull 并且 git push到远程仓库?

直接使用git pull 无法拉取远程仓库版本信息，因为是本地和远程仓库是不同的独立仓库，要拉取必须使用 --allow-unrelated-histories参数允许合并无关历史

如：

```
git pull origin main --allow-unrelated-histories
```

合并后解决可能的冲突，再 `git push` 即可。

3. 请解释一下git 的相关概念、工作区、暂存盘、版本库、远程仓库，介绍它们的关系是什么？操作流程是什么?

   git概念：Git 是一个开源的**分布式版本控制系统**，用于追踪文件变更、协作开发、版本回退等。

   工作区：正在编辑文件的目录，就是你的项目文件夹

暂存盘：临时区域，用于存放你准备提交（commit）的文件变更。

版本库：本地仓库，保存项目完整历史记录的地方，位于项目根目录下的 `.git` 文件夹中。

远程仓库：托管在服务器上的 Git 仓库，如 GitHub、Gitee、GitLab 等平台上的项目仓库。

关系：

- **工作区 → 暂存区**：`git add`
- **暂存区 → 本地版本库**：`git commit`
- **本地版本库 ↔ 远程仓库**：`git push` / `git pull` / `git fetch`
- **远程仓库 → 本地版本库 → 工作区**：`git clone`（首次拉取整个项目）

流程：

```
# 1. 在工作区修改文件（如：edit index.html）

# 2. 查看当前状态（可选）
git status

# 3. 把修改添加到暂存区
git add index.html
# 或添加所有改动：git add .

# 4. 提交到本地版本库
git commit -m "描述你的修改，例如：更新首页布局"

# 5. 推送到远程仓库（如 origin 的 main 分支）
git push origin main
```

4. 假如现在项目有一个报错，通过报错提示我定位到具体的文件，但是我现在弄清楚这个错误是谁导致的，git怎么做？

使用命令`git blame 文件名`

显示指定文件的每行代码最后一次修改的提交信息（作者、时间、代码内容）。

5. 我想知道工作区和暂存盘的区别？git使用什么命令合适？

   工作区：正在编辑文件的目录，就是你的项目文件夹

   暂存盘：一个临时区域，用于存放你准备提交（commit）的文件变更。

   使用命令：

```
git diff                    # 查看工作区与暂存区差异
git diff --cached       # 查看暂存区与版本库差异
git diff HEAD             # 查看工作区与版本库差异
```

6. 我在做项目实现一个功能，我试了好几个方案，都不理想，最后我权衡下来，觉得还是方案2可行性高，打算从方案2继续研究下去。请问你接下来git怎么操作比较合适？

   多方案选择回溯，应使用命令`git reset`回退到指定提交：

```
git reset --hard 版本号   # 彻底回溯到方案2
git reset --soft 版本号   # 保留后续修改
```

7. 我在项目因为我对项目有洁癖，被我不小心删除了一个config文件，请问我怎么找回来？

   工作区撤销修改：

   - git status 查看修改的文件
   - 使用命令恢复：`git checkout -- 被恢复的文件`

   暂存盘撤销修改：

   - `git reset HEAD 被恢复的文件`
   - `git checkout -- 被恢复的文件`

   已提交：

```
git checkout 版本号 -- 文件   # 从历史版本恢复
```

8. 我有个项目打算使用git做分布式管理， 但是当我提交一个版本的时候居然不能提交，请问有几种可能导致不能提交?
    - **本地分支未关联远程分支**：`git branch --set-upstream-to=origin/远程分支名 本地分支名`
    - **提交信息不规范**（某些仓库要求符合conventional commit）
    - **文件大小超限**（如GitHub限制100MB）
    - **网络代理问题**
    - **远程分支保护策略**（如main分支需PR才能合并）
    - **无写权限** → 检查远程仓库权限或 SSH key
9. 我有个项目是多人协作的，我完成了一个功能，组长催促我赶紧上传到远程，我于是保存版本后，开始使用git push上传，但不能成功上传，请问有几种可能导致不能推送到远程?
    - **本地分支未关联远程分支**：`git branch --set-upstream-to=origin/远程分支名 本地分支名`
    - **提交信息不规范**（某些仓库要求符合conventional commit）
    - **文件大小超限**（如GitHub限制100MB）
    - **网络代理问题**
    - **远程分支保护策略**（如main分支需PR才能合并）
    - **无写权限** → 检查远程仓库权限或 SSH key
10. 练习成语接龙游戏，自己找搭档配合练习， 不想练就把整个过程文字描述一下。
    - git pull拉取远程仓库最新的提交和分支信息，
    - 在本地进行修改添加文件内容，
    - 使用git push提交至远程仓库，
    - **提交不了**则使用git stash保存工作区内容，使用git pull拉取最新提交版本信息，在本地解决冲突后，git add保存解决冲突后的版本，git commit提交一个版本，最后git push到远程仓库

```
git pull                                    # 拉取远程更新（可能冲突）
git stash                                   # 保存本地未提交修改
git pull                                    # 成功拉取
git stash pop                               # 恢复并合并，手动解决冲突
git add .                                   # 标记冲突已解决
git commit -m "fix: resolve conflict"
git push origin main                        # 推送
```

11. 练习分支管理、查看分支、删除分支、添加分支、切换分支、创建切换分支

操作	命令
查看本地分支	`git branch`
查看详细分支	`git branch -v`
查看所有分支	`git branch -a`
创建分支	`git branch dev`
切换分支	`git switch dev` 或 `git checkout dev`
创建+切换	`git switch -c dev` 或 `git checkout -b dev`
合并分支	`git merge dev`
删除分支（安全）	`git branch -d dev`
强制删除分支	`git branch -D dev`

> ⚠ **注意**：
>
> - 不能删除当前所在分支。

- `-D` 是强制删除，即使未合并也可删，**不适用于远程分支**（远程分支需 `git push origin --delete dev`）。

12. 练习抓取远程分支、远程修改默认分支

```
git checkout -b dec

git push origin dev

git switch -c bug origin
```

```
$ git checkout -b dev
Switched to a new branch 'dev'

NINGXIU@MSI MINGW64 /a/Desktop/git-study/hello/hello9-17 (dev)
$ git branch -a
* dev
  main
  remotes/origin/HEAD -> origin/main
  remotes/origin/main

NINGXIU@MSI MINGW64 /a/Desktop/git-study/hello/hello9-17 (dev)
$ git push origin dev
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Powered by GITEE.COM [1.1.5]
remote: Set trace flag 0742f5bc
remote: Create a pull request for 'dev' on Gitee by visiting
NINGXIU@MSI MINGW64 /a/Desktop/git-study/hello/hello9-17 (dev)
$ git switch -c bug origin/bug
fatal: invalid reference: origin/bug

NINGXIU@MSI MINGW64 /a/Desktop/git-study/hello/hello9-17 (dev)
$ git switch -c bug origin/main
branch 'bug' set up to track 'origin/main'.
Switched to a new branch 'bug'

NINGXIU@MSI MINGW64 /a/Desktop/git-study/hello/hello9-17 (bug)
$ git branch -a
* bug
  dev
  main
  remotes/origin/HEAD -> origin/main
  remotes/origin/dev
  remotes/origin/main
```

13. 再练习一遍git stash的操作

1.工作区修改文件（不要添加暂存和保存版本）

2.远程仓库修改文件并提交版本

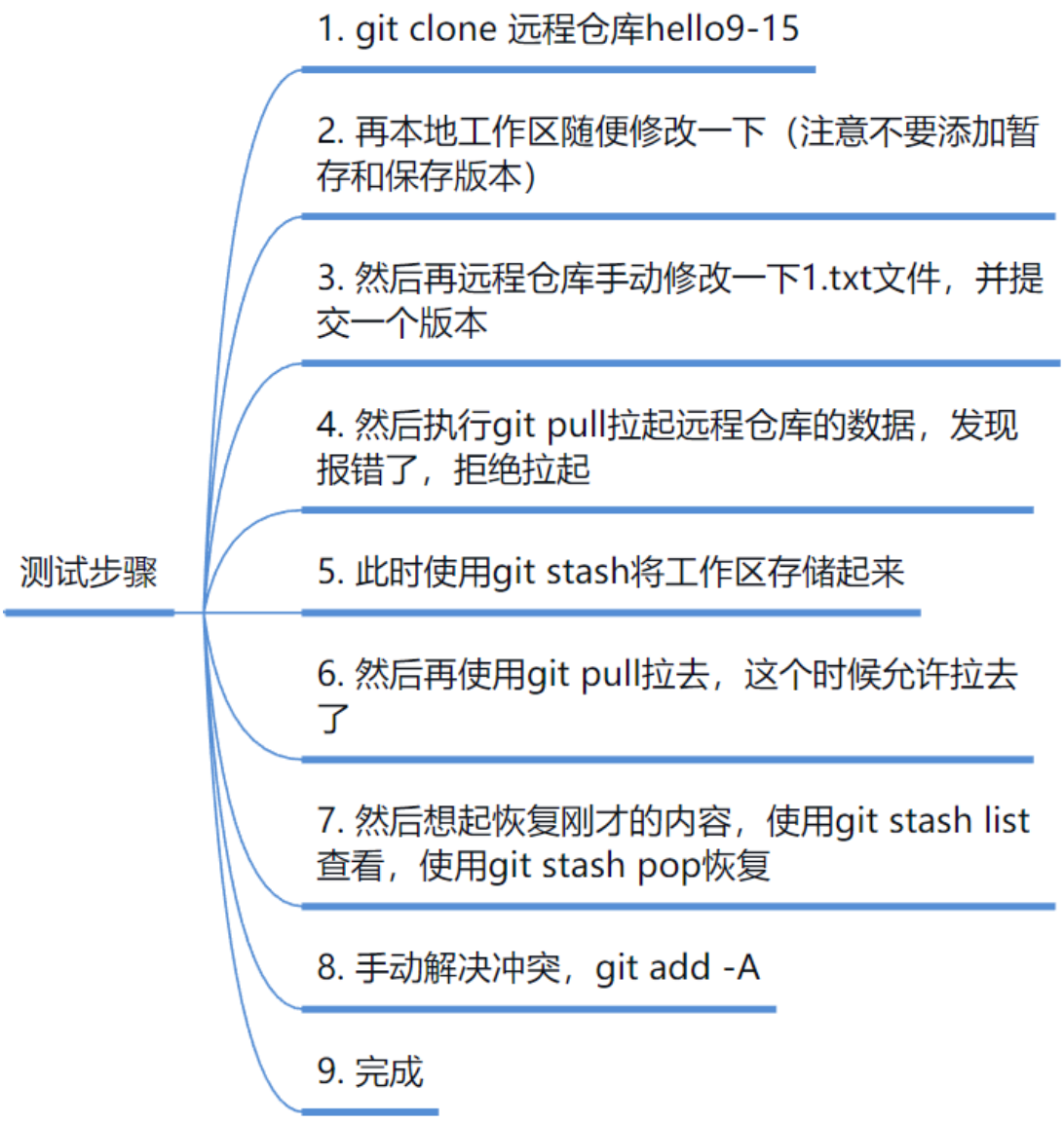
3.执行git pull报错，拒绝拉取

4.执行git stash 存储工作区

5.执行git pull，允许拉取

6.使用git stash list 查看，使用git stash pop恢复

7.手动解决冲突，git add -A



14. git branch -d 和 git branch -D的区别?

`git branch -d 分支名`：删除已合并的分支

`git branch -D 分支名`：**强制删除**（无论是否合并），可以删除远程分支

> ❗ **两者都不能删除当前所在分支！**
>
> ❗ **远程分支删除命令是：** `git push origin --delete dev`

15. .gitignore文件有什么作用?

    .gitignore 作用：上传忽略名单

    可以在.gitignore文件中添加文件夹，指定文件或者指定类型的文件

```
# 通用模板示例
*.log
.DS_Store
node_modules/

# 排除特定文件夹但保留必要文件
!/logs/important.log

# 环境变量文件
.env
*.secret
```

16. git log和 git reflog的区别?

    `git log`：查看历史版本到当前版本的日志（完整提交信息）

    `git reflog`：查看所有历史版本信息（简写哈希+操作动作）

    查看的是一个短的版本号，例如 da213db

二、 今晚默写和语音题

1. git仓库中的https和ssh协议的区别?（默写+语音朗读）
   - **https协议**在推送的时候会弹出登录框，输入仓库托管网站（gitee或者github等）的账户和密码，输入成功后，我们的操作系统会自动记住登录状态，下一次推送不需要再输入账号和密码
   - **ssh协议**需要自己生成公钥，并且和仓库托管网站登录账户进行绑定，即可。绑定成功后，也是可以免密码登录推送的。

2. 请介绍推送本地仓库到远程的过程？（默写+语音朗读）

```
# 添加暂存盘
git add -A

# 提交一个版本
git commit   -m “描述文字”

# 添加远程仓库别名
git remote add origin 仓库地址

# 推送到远程仓库
git push -u origin main
```

答案在下面的网址
http://47.110.66.96/helper/guide/git.html#%E5%A6%82%E4%BD%95%E5%88%9B%E5%BB%BA%E5%92%8C%E5%90%88%E5%B9%B6%E5%88%86%E6%94%AF

三、 明天早上默写和语音题

1. git rebase 和 git merge的区别？（默写+面试提问）

   `git rebase` **和** `git merge` **都是用来“整合分支”的工具，但它们的实现方式、历史记录结构、适用场景完全不同。**

   `merge` **是“合并”，保留完整历史分支结构；**
   `rebase` **是“变基”，把你的提交“挪到”目标分支最新提交之后，让历史线性整洁。**

2. 如何解决提交版本产生冲突？（默写+面试提问）

   > 1. 多人同时修改同一文件的同一区域
   > 2. 执行 `git pull` / `git merge` / `git rebase` 时
   > 3. 分支合并时存在不兼容的修改

解决：

- 使用`git status`查看冲突文件
- 手动打开文件解决冲突
- 使用git add标记已解决的冲突文件
- 使用git commit 提交修改
- 推送到远程仓库：git push origin main

3. 请介绍推送本地仓库到远程的过程？（默写+面试提问）

```
# 添加暂存盘
git add -A

# 提交一个版本
git commit   -m “描述文字”

# 添加远程仓库别名
git remote add origin 仓库地址

# 推送到远程仓库
git push -u origin master
```

4. 请介绍git多人远程协作的流程？（默写+面试提问）

    1. **克隆远程仓库**

首先，每个开发者需要将远程仓库克隆到本地：

```
git clone <远程仓库URL>
```

    2. **创建分支**

为了隔离开发工作，开发者应基于主分支（如 main 或 master）创建新分支：

```
git checkout -b <分支名>
```

    3. **开发与提交**

在本地分支上进行开发，完成后提交更改：

```
git add .

git commit -m "提交信息"
```

4. **推送分支**

将本地分支推送到远程仓库：

```
git push origin <分支名>
```

5. **创建 Pull Request (PR)**

在远程仓库（如 GitHub、GitLab）上，基于推送的分支创建 PR，请求将更改合并到主分支。

6. **代码审查**

团队成员审查 PR，提出修改建议。开发者根据反馈更新代码，并推送新的提交。

7. **合并 Pull Request (PR)**

审查通过后，将 PR 合并到主分支。

8. **同步主分支**

合并后，开发者应拉取最新的主分支到本地，保持同步：

```
git checkout main

git pull origin main
```

9. **删除已合并的分支**

合并完成后，可以删除远程和本地的已合并分支：

```
git push origin --delete <分支名>

git branch -d <分支名>
```

10. **处理冲突**

如果多人修改了同一文件，可能会产生冲突。解决冲突后，标记冲突已解决并提交：

```
git add <冲突文件>

git commit -m "解决冲突"
```

11. **持续集成**

许多团队使用 CI/CD 工具（如 `Jenkins`、`GitHub Actions`）自动测试和部署代码，确保每次合并后的代码质量。

12. **定期同步**

开发者应定期拉取主分支，保持本地代码与远程仓库同步，减少冲突。

总结:

多人协作的关键在于:

使用分支隔离开发

通过 PR 进行代码审查

定期同步主分支

及时解决冲突