**Git 是一个分布式版本控制系统，它允许开发者对代码进行版本控制**，以下是一些常用的 Git 命令及其使用示例：

**Git工作区、暂存区和版本库之间的区别是什么？**

工作区（Working Directory）：当前文件所在的目录，可以编辑和修改代码。

暂存区（Staging Area/Index）：已标记为待提交的文件区域，通过git add命令将文件从工作区移到暂存区。

版本库（Repository）：Git管理的代码库，包含所有历史提交和分支，通过git commit将暂存区的内容提交到版本库。’'

以下是一些常用的 Git 命令及其使用示例：

**初始化仓库**：

git init

这会在当前目录创建一个新的 Git 仓库。

**克隆仓库：**

git clone [仓库地址]

例如，克隆一个 GitHub 上的仓库：

git clone https://github.com/username/repository.git

**添加文件到暂存区：**

git add [文件名]

添加当前目录下的所有文件：

git add .

**提交更改：**

git commit -m "提交信息"

例如，提交当前暂存区的更改：

git commit -m "Add new feature"

**查看提交历史：**

git log

这会显示仓库的提交历史。

**查看当前状态：**

git status

这会显示当前工作目录和暂存区的状态。

**查看差异：**

git diff

这会显示工作目录与暂存区的差异。

**回退到上一个提交：**

git reset --hard HEAD^

这会将 HEAD 指针移动到上一个提交，并丢弃当前提交。

**查看远程仓库：**

git remote -v

这会列出所有远程仓库的 URL。

**添加远程仓库：**

git remote add origin [仓库地址]

例如，添加一个远程仓库：

git remote add origin https://github.com/username/repository.git

**拉取远程仓库的更改：**

git pull [远程名] [分支名]

例如，从远程仓库的 master 分支拉取更改：

git pull origin master

**推送到远程仓库：**

git push [远程名] [分支名]

例如，推送到远程仓库的 master 分支：

git push origin master

**创建新的分支：**

git branch [分支名]

例如，创建一个名为 feature 的新分支：

git branch feature

**切换分支：**

git checkout [分支名]

切换到 feature 分支：

git checkout feature

**合并分支：**

git merge [分支名]

合并 feature 分支到当前分支：

git merge feature

**解决冲突：**

当合并时出现冲突，需要手动解决冲突，然后使用：

git add [冲突解决后的文件]

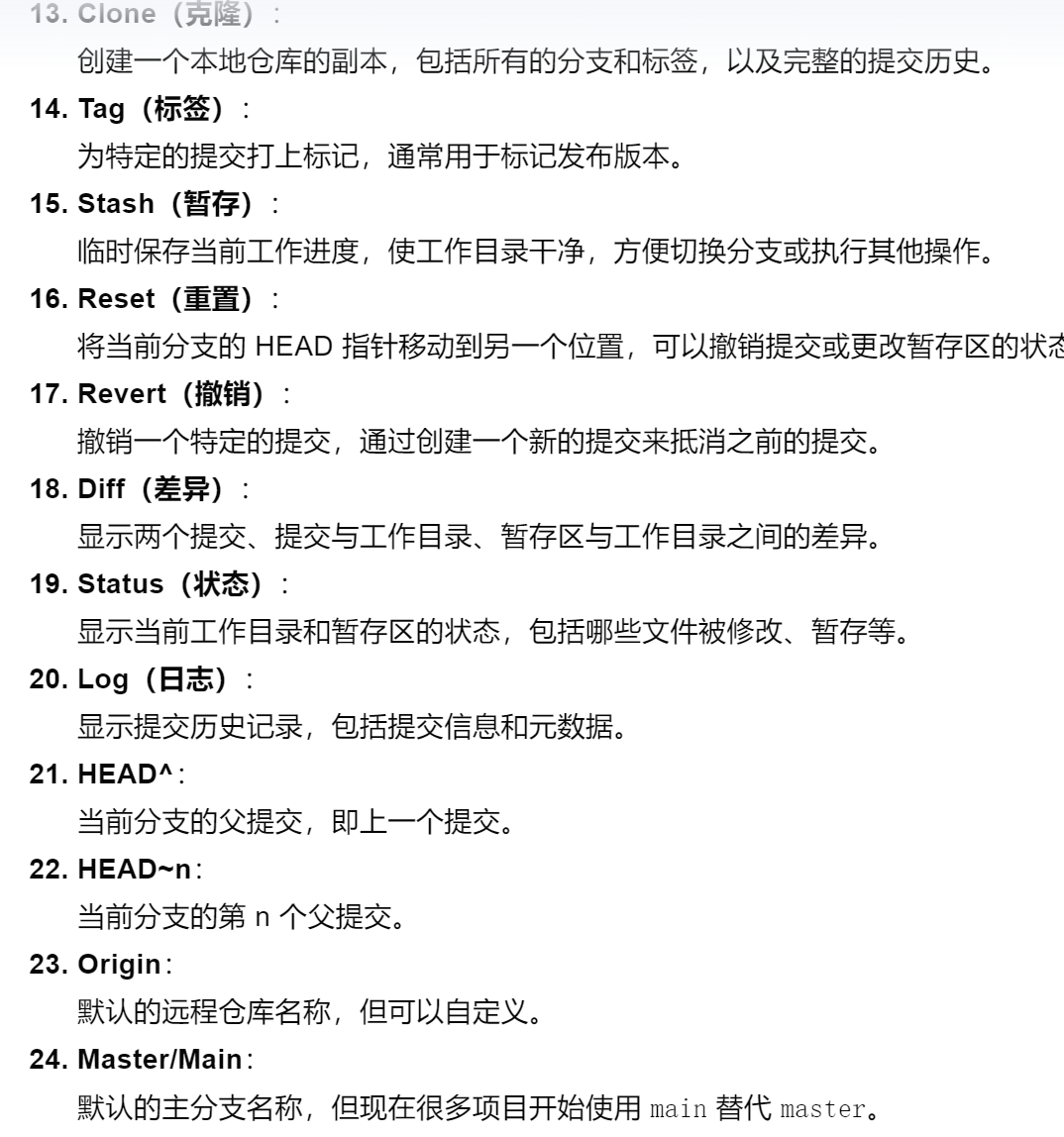
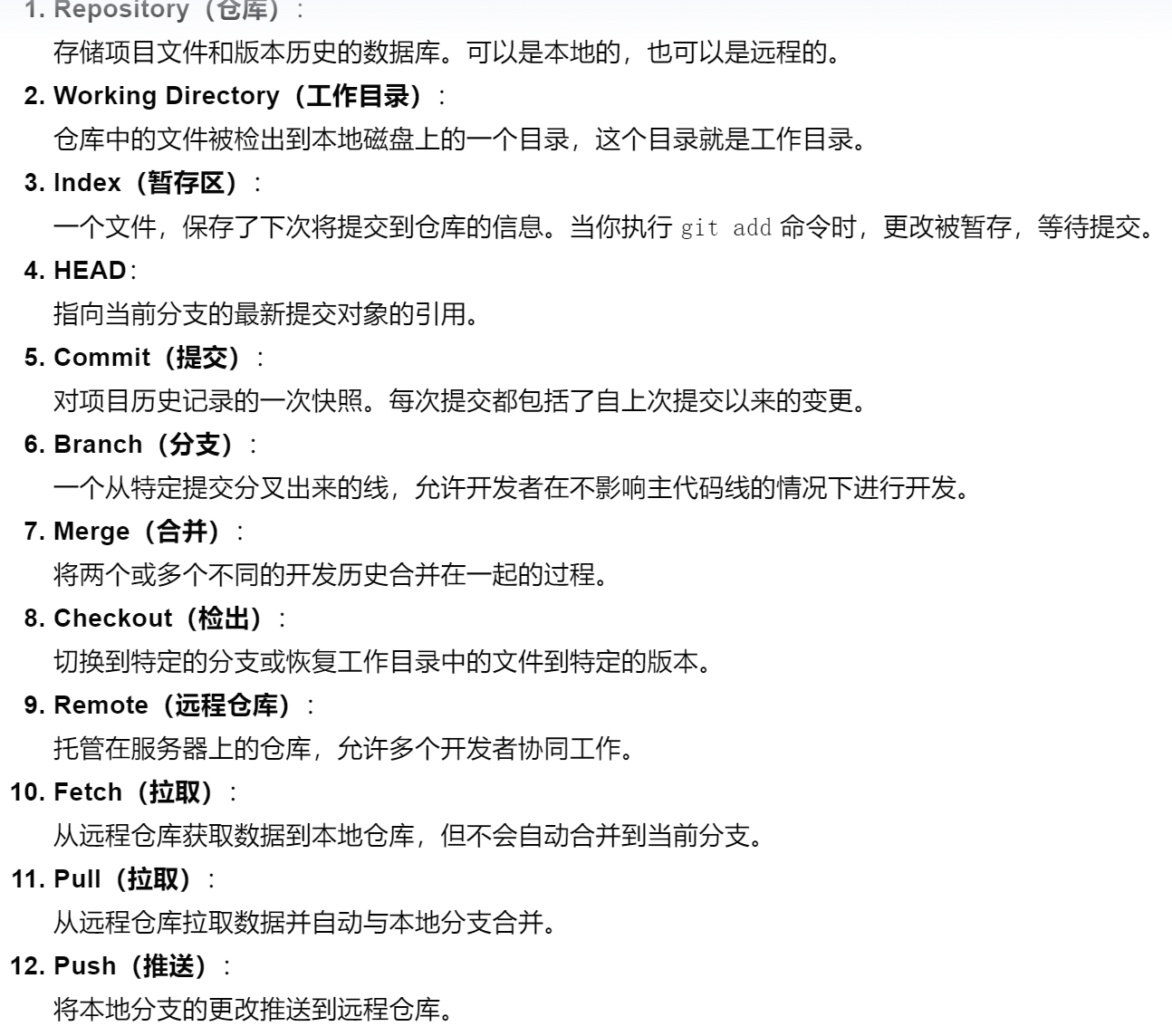
git commit

**删除分支：**

git branch -d [分支名]

删除 feature 分支：

git branch -d feature

\