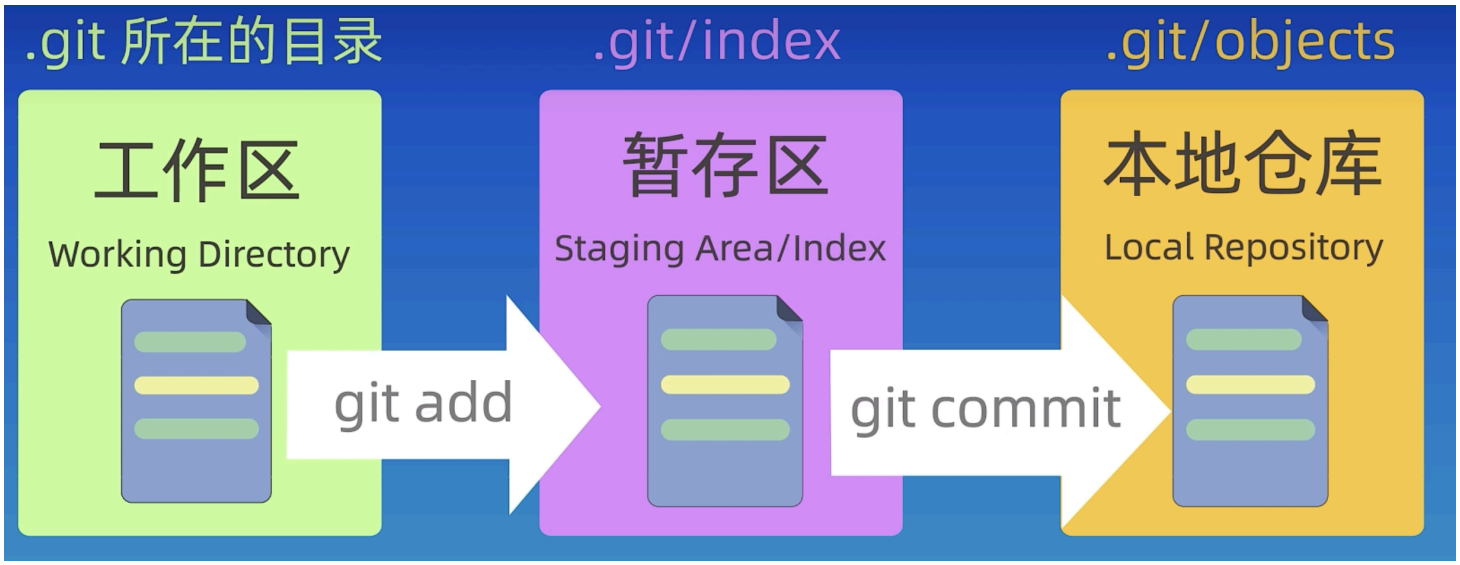
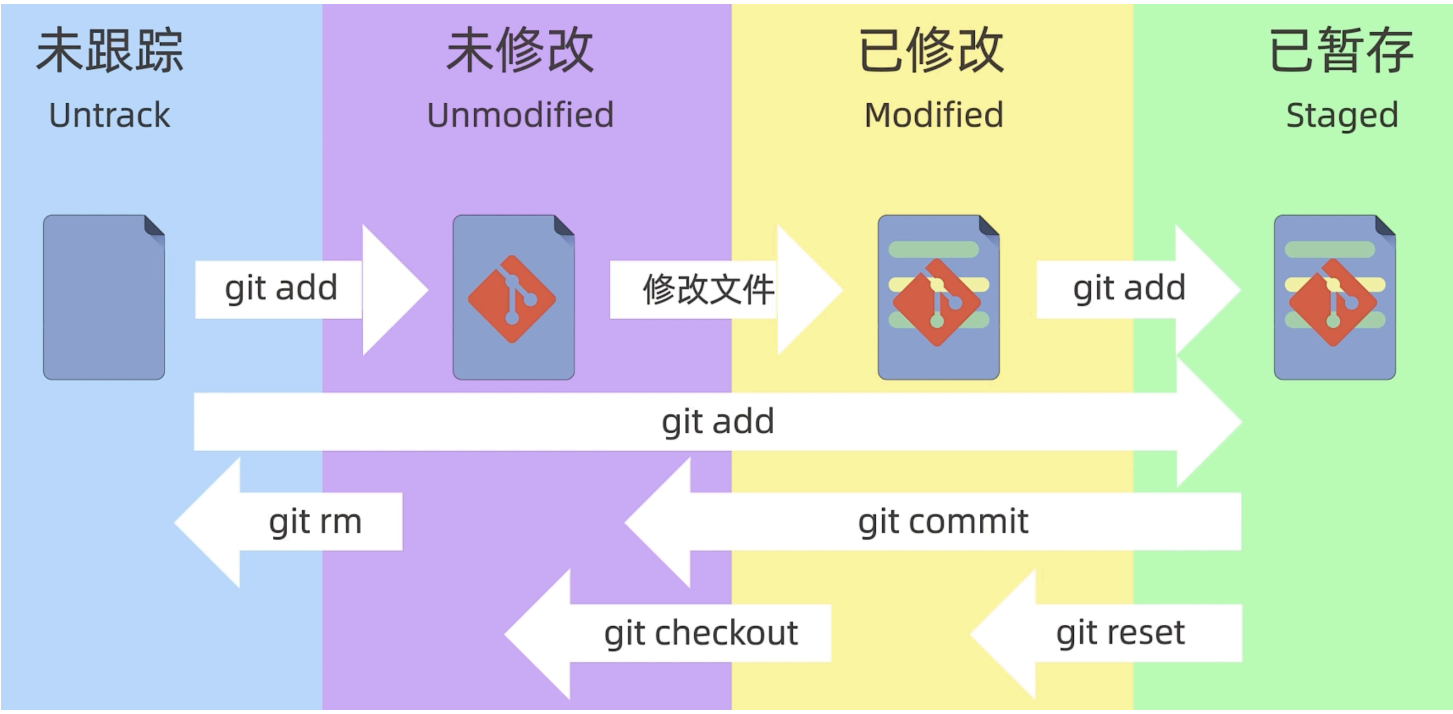
git学习过程

1. 新建仓库

使用 `git init` 创建git仓库，会在当前文件夹下创建仓库
使用 `git init my_repo` 创建git仓库，会新建my_repo文件夹并在其下创建仓库
新建仓库会在文件夹中生成 **.git文件夹**

2. 创建的.git文件夹



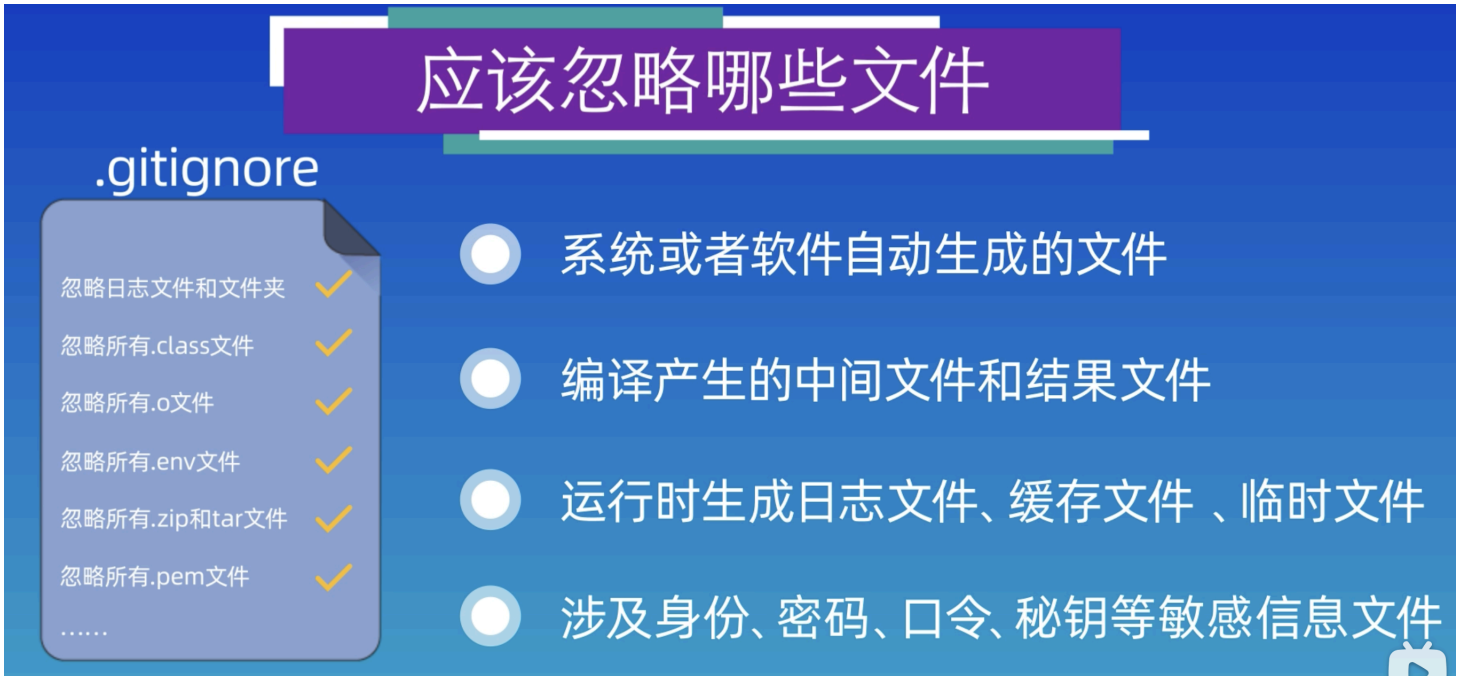


3. git命令

- 查看仓库状态: `git status`
- 添加到暂存区
  - 单文件: `git add <file>`
  - 使用目录: `git add .` (目录里的所有文件)
  - 使用通配符: `git add *.txt` （所有以txt结尾的文件）
- 取消暂存: `git rm --cached <file>`
- 提交: `git commit -m "提交信息"` *仅会提交* **暂存区里的文件**
- 查看提交日志: `git log`
  - 简洁版: `git log --online`
- 回退版本: `git reset`
  - `--soft` **保留** *工作区* 和 *暂存区* 的 **所有修改内容**
  - `--hard` **丢弃** *工作区* 和 *暂存区* 的 **所有修改内容**
  - `--mixed` **保留** *工作区* 和 **丢弃** *暂存区* (默认参数)
  - 回到指定版本: `git reset --xxx <版本号>`
  - 回到上一版本: `git reset --xxx HEAD^`
- 比较差异: `git diff` (HEAD~n 表示当前版本的上n版本，n从2开始，不写表示上一个版本)
  - 默认比较 ***工作区*** 和 ***暂存区*** 之间的差异
  - `HEAD` 比较 ***工作区*** 和 ***版本库*** 之间的差异
  - `--cached` 比较 ***暂存区*** 和 ***版本库*** 之间的差异
  - `<版本号1> <版本号2>` 比较 ***版本1*** 和 ***版本2*** 之间的差异

    - `<branch <branch>` 比较 ***分支*** 之间的差异
    - `git diff xxx <file>` 表示仅比较 ***指定文件*** 之间的差异
- 删除文件，删除后不要忘记提交
    - `git ls-files` 查看 *暂存区* 的文件内容
    - `rm <file>; git add <file>` 先从 *工作区* 删除文件，然后再从 *暂存区* 删除文件
    - `git rm <file>` 把文件从 *工作区* 和 *暂存区* 同时删除
    - `git rm --cached <file>` 把文件从 *暂存区* 删除，但保留在当前 *工作区* 中
    - `git rm -r *` 递归删除某个目录下的所有子目录和文件

4. .gitignore文件



5. SSH配置和克隆仓库

- 生成SSH Key: `ssh-keygen -t rsa -b 4096`
    - 私钥文件: id_rsa
    - 共钥文件: id_rsa.pub
- 克隆仓库: `git clone <repo-address>`
- 推送更新内容: `git push <remote> <branch>:<branch>`
- 拉取更新内容: `git pull <remote> <branch>:<branch>`
    - `<remote> <branch>:<branch>` :<远程仓库名> <远程分支名>:<本地分支名> (可不写)
- 查看远程仓库管理情况: `git remote -v`
- 关联另一个远程仓库: `git remote add <repo-address>`

6. 分支基本操作

- 查看分支列表: `git branch`
- 创建分支: `git branch <branch>`
- 切换分支: `git switch <branch>` (推荐)
  - `git checkout <branch>` (也可切换分支)
- 合并分支: `git merge <branch>` (要先切换到要合并到的分支，表示将 `<branch>` 分支合并到当前分支)
- 删除分支:
  - 已合并: `git branch -d <branch>`
  - 未合并: `git branch -D <branch>`

7. 解决合并冲突

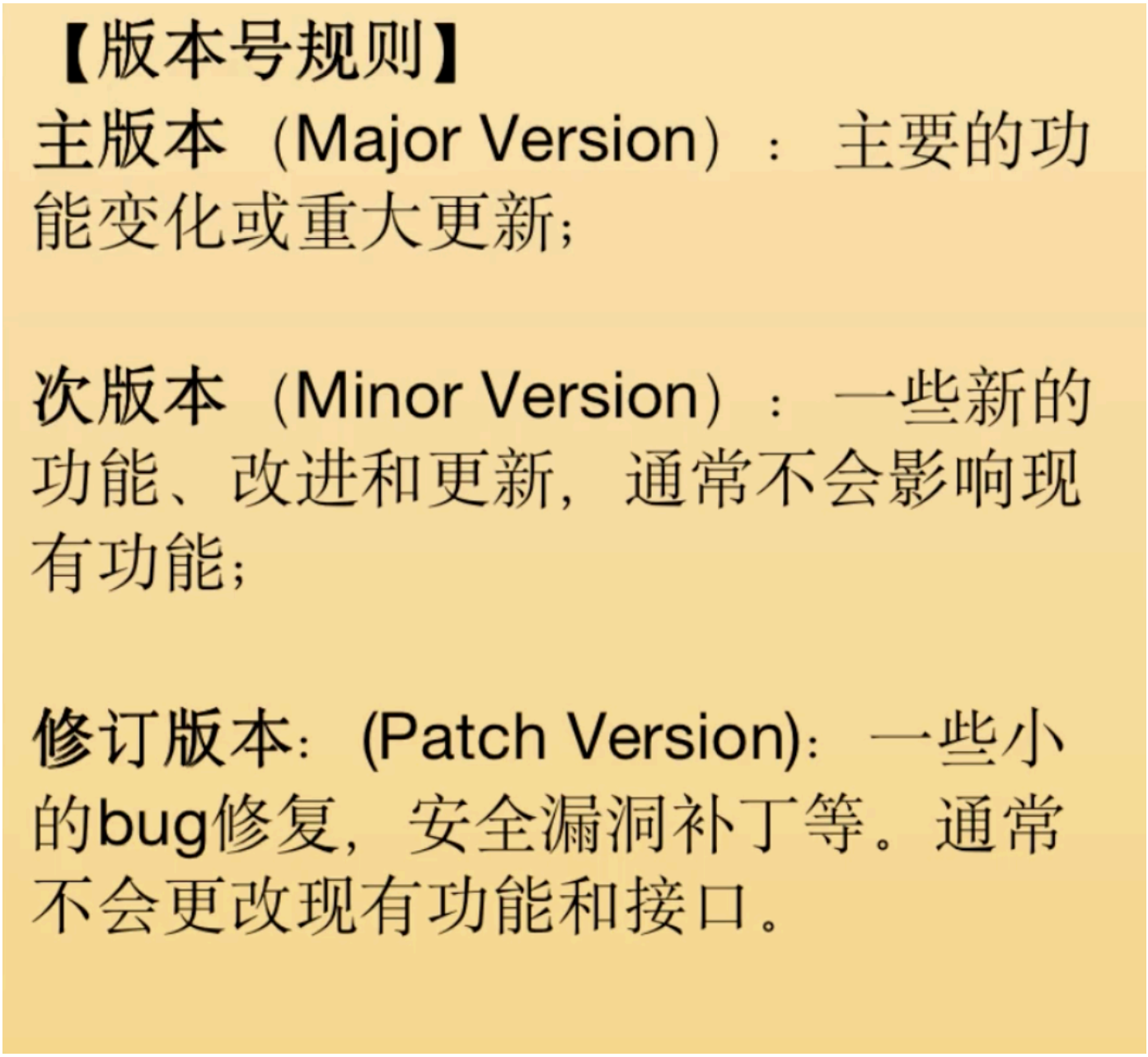
- 两个分支未修改同一个文件的同一处位置: Git自动合并
- 两个分支修改了同一个文件的同一处位置: 产生冲突conflict
  - 解决办法:
    - step1 - 手工修改冲突文件，合并冲突内容
    - step2 - 添加暂存区 `git add <file>`
    - step3 - 提交修改 `git commit -m "<message>"`
      中止合并: `git merge --abort` (当不想继续执行合并操作时中止合并过程)

8. 回退和变基rebase

- 切换到某分支并回退到指定版本: `git checkout -b <branch> <branch_id>`
- 将当前分支变基到目标分支: `git rebase <branch>`
- Merge (***一般将自己的分支merge到公共分支***)
  - 优点: 不会破坏原分支的提交历史，方便回溯和查看
  - 缺点: 会产生额外的提交节点，分支图比较复杂
- Rebase (***一般在自己的分支使用，避免在共享分支使用***)
  - 优点: 不会新增额外的提交记录，形成线性历史，比较直观和干净
  - 缺点: 会改变提交历史，改变了当前分支branch out的节点

9. 工作流模型

- GitFlow模型:



- **main** 主分支 (永久)
- **develop** 开发分支 (永久)
- **hotfix** bug热修复分支，从**main branch**分离出来
- **各种功能分支** 从**develop branch**分离出来
- **release** 测试分支 (永久)