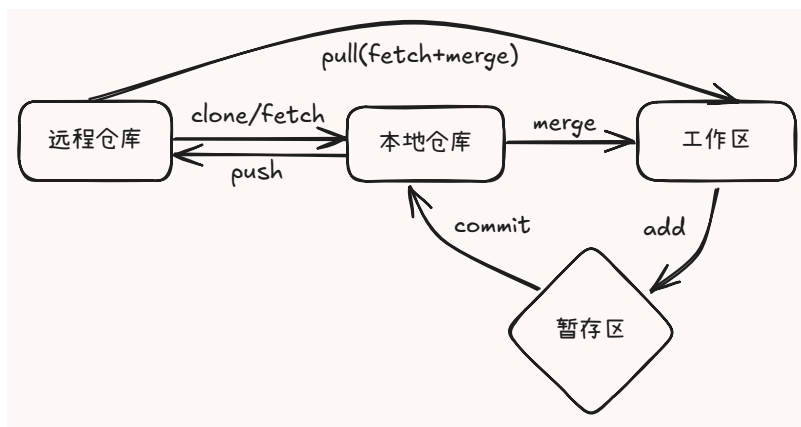


| Git

| 1 工作流程



| 2 基本配置

1. 设置用户名和邮箱

```
git config --global user.name "your name"
git config --global user.email "your email"
```

2. 查看用户名和邮箱

去掉↑的 "xxxx" 部分

3. 创建指令别名

在用户目录^[1]下创建 `.bashrc` 文件

- 通过命令 `touch ~/.bashrc` 来创建文件
- 打开 `.bashrc`, 输入

```
#用于输出git提交日志
alias git-log='git log --pretty=oneline --all --graph
--abbrev-commit'
#用于输出当前目录所有文件及基本信息
alias ll='ls -al'
```

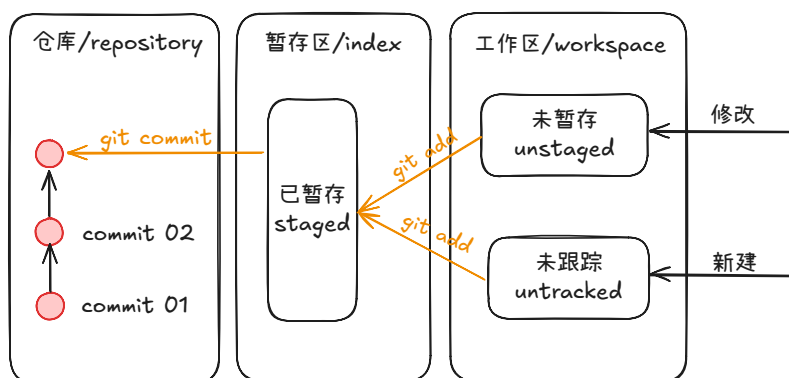
- 执行命令 `source ~/.bashrc`

| 3 创建本地仓库

1. 创建新目录并打开git bash
2. 通过 `git init` 命令初始化新的本地仓库(产生 `.git` 文件夹)

| 4 基础操作指令

用于修改工作目录^[2]



| 4.1 git add

将文件从工作区加入暂存区

- 可以通过 `git add 文件名` 添加要上传的文件
- 可以通过 `git add .` 上传所有文件



坑

`git add .` 不上传空文件夹

在空文件夹中创建 `.gitkeep` 文件来使git上传

| 4.2 git commit

将文件提交到仓库(产生版本)

- 通过 `git commit -m "消息"` 提交(`-m` 指提交变更消息)

| 4.3 git status

查看仓库状态(暂存区和工作区)

| 4.4 git log

查看仓库指令历史

参数:

- `--all` 显示所有分支
- `--pretty=oneline` 将提交信息显示为一行
- `--abbrev-commit` 使得输出的commitId更简短
- `--graph` 以图的形式显示

显示:

- `1b37cbd (HEAD -> dev01)` 说明当前工作区在dev01分支上

| 4.5 git reset

回退版本

使用 `git reset --hard commitID`

- commitID用 `git log` 查看

| 4.6 git reflog

查看版本回退记录,用来恢复回退的版本

| 4.7 添加文件忽略列表

- 创建 `.gitignore` 文件
- 修改 `.gitignore` 文件

| 5 分支命令

| 5.1 git branch

使用 `git branch 分支名` 创建新分支

使用 `git branch` 查看分支(多用 `git log` 替代)

使用 `git branch -d 分支名` 删除分支(`-D` 不做检查,强制删除)

使用 `git branch -vv` 查看本地与远程的关联关系

工作区只能为一个分支服务

| 5.2 git checkout

使用 `git checkout 分支名` 切换分支

使用 `git checkout -b 分支名` 创建并切换分支 #Usually

| 5.3 git merge

使用 `git merge 分支名` 将分支合并到master

eg.分支 *dev* 与 *master* 产生冲突

Git bash报错

```
Auto-merging file01.txt
CONFLICT (content):Merge conflict in file01.txt
Automatic merge failed;fix conflicts and then commit the
result.
```

文件内容

File01

```
<<<<<<< HEAD
update count=3
=====
update count=2
>>>>>> dev
```

改为

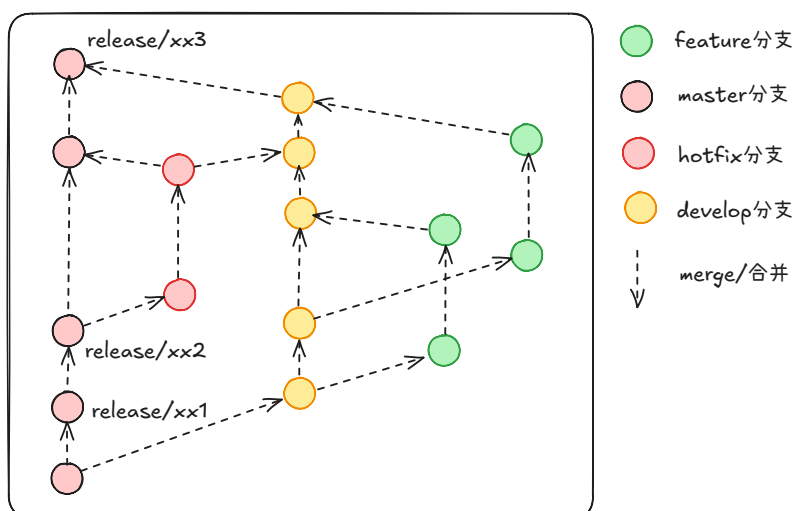
File01_changed

```
update count=4
```

再通过 `git add .` 与 `git commit` 提交即可

| 5.4 分支使用原则与流程

- `master` 用于线上
- `develop` 开发时要合并进的分支,最终合并进 `master`(`develop` 一般不删)
- `feature` 用于不同开发需求,完成后合并进 `develop`
- `hotfix` 用于修复bug,完成后合并进 `master`



| 6 远程仓库

在创建完本地仓库后

创建远程仓库时不能添加文件,否则本地push到远端出错

| 6.1 git remote

- 通过 `git remote add 远端名称 远程仓库路径` 建立连接
远端名称一般命名为 `origin`
- 通过 `git remote` 查看远程仓库

| 6.2 git push

- 第一次push时
使用 `git push 远端名称 本地分支名:远端分支名`
通过参数 `--set-upstream(-u)` 进行绑定
- 绑定后直接使用 `git push` 即可

| 6.3 git clone

通过 `git clone 远程链接 (本地仓库名)` 克隆远程仓库
本地仓库名可以自动生成或指定

| 6.4 抓取和拉取

1. `git fetch`

- 抓取不进行合并,把远端仓库抓取到本地,但是不影响本地分支
- 通过 `git merge` 进行本地合并

2. `git pull`

- 抓取并合并

| 6.5 解决合并冲突

同本地冲突修改

| 5.3 git merge

使用 `git merge 分支名` 将分支合并到master
eg.分支 `dev` 与 `master` 产生冲突

Git bash报错

```
Auto-merging file01.txt
CONFLICT (content):Merge conflict in file01.txt
Automatic merge failed;fix conflicts and then commit the
result.
```

文件内容


File01

```
<<<<<<< HEAD
update count=3
=====
```

```
update count=2
```

```
>>>>>> dev
```

改为

 **File01_changed**

```
update count=4
```

再通过 `git add .` 与 `git commit` 提交即可

-
1. 用户目录: 就是user目录下的具体用户目录 ↩
 2. 工作目录: 除了 `.git` 文件夹的其他内容 ↩