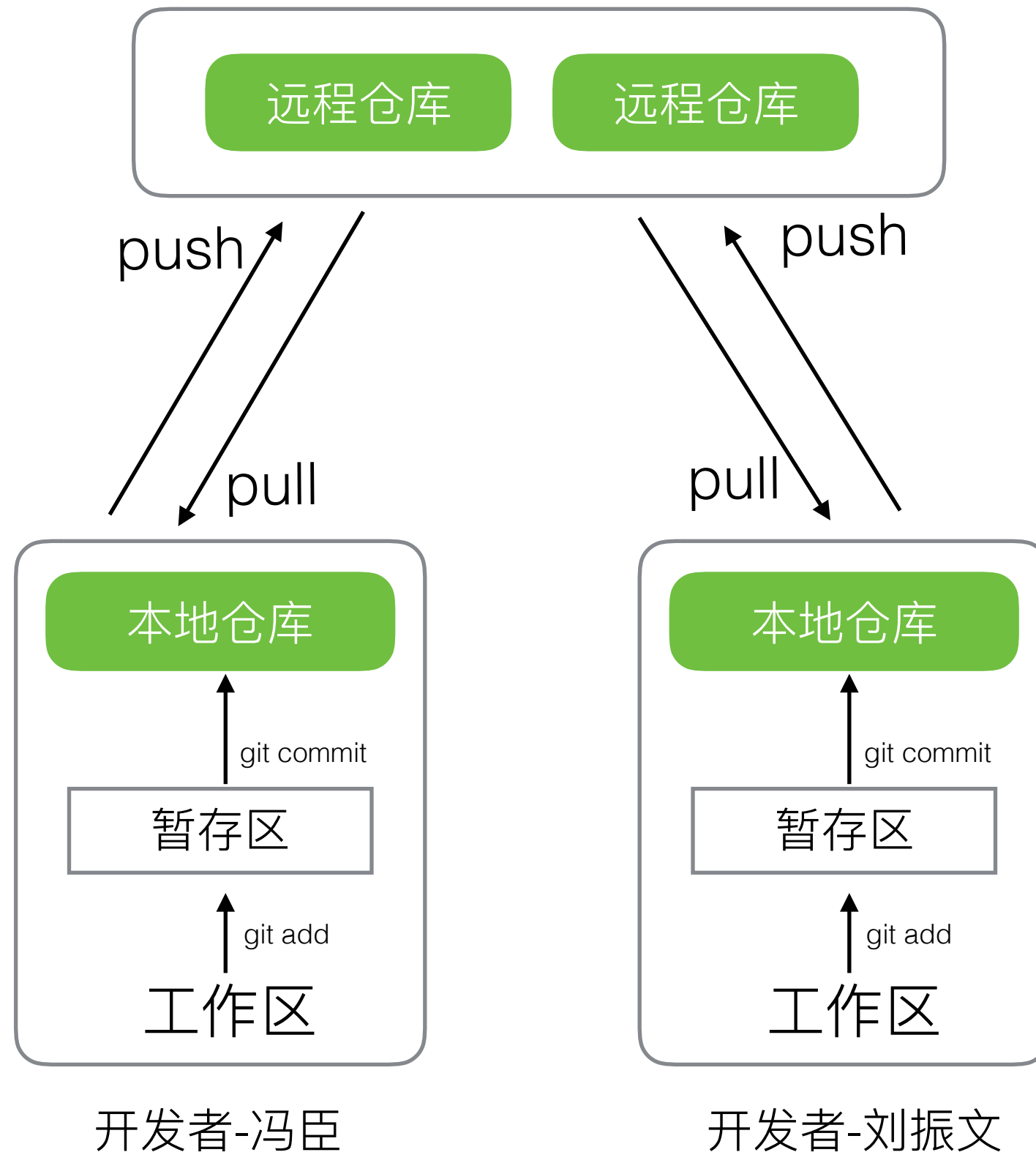


# Git版本管理系统

- 分布式
- 分支
- 随时保存更新记录
- 不用再保存一个 .bak 文件啦

# 运行流程



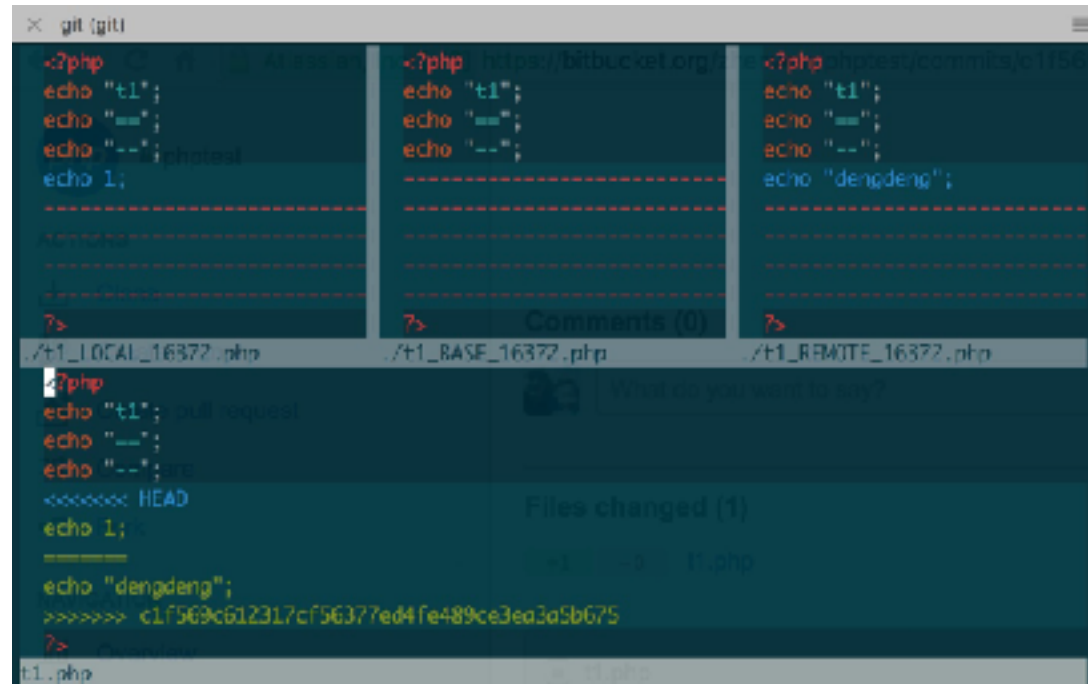
# Git常用命令

- 新建代码仓库
  - `git init` (在当前目录新建一个git仓库)
  - `git init [project-name]` (新建一个目录并初始化git仓库)
  - `git clone [url]` (下载一个仓库)
- 配置 (Git的设置文件为.gitconfig, 用户主目录下是全局配置, 项目目录下是项目配置)
  - `git config --list` (显示当前项目的配置)
  - `git config -e [--global]` (编辑Git配置文件, 也可以找到对应的.gitconfig文件去编辑)
- 文件操作
  - `git add [file]` (添加文件到暂存区), `git add .` (添加所有文件到暂存区)
  - `git commit -m [message]` (提交暂存区的文件到本地仓库)
  - `git rm [file]` (删除文件并放入暂存区)
  - `git rm --cached [file]` (从版本库中移除文件, 文件会保留在工作目录中)
  - `git mv [file-original] [file-renamed]` (修改文件名, 放入暂存区)

- 分支
  - git branch (列出本地分支, 加上 -r 参数列出远程分支, -a 参数列出本地和远程分支)
  - git branch [name] (新建分支, 停留在当前分支)
  - git branch -d [name] (删除分支)
  - git checkout -b [name] (新建分支并切换到此分支, 去掉 -b 参数就是切换到此分支)
  - git merge [branch] (合并分支到当前分支)
- 远程同步
  - git remote -v (显示所有远程仓库)
  - git remote add [shortname] [url] (增加一个远程仓库并命名)
  - git fetch [remote] (下载远程仓库的所有变动)
  - git pull [remote] [branch] (拉取远程仓库分支的更新)
  - git push [remote] [branch] (推送本地仓库分支修改到远程)
  - git pull = git fetch + git merge

# 冲突解决

## 1. git mergetool [file]



The screenshot shows the git mergetool interface with three panels. The left panel shows the local version of t1.php, the middle panel shows the base version, and the right panel shows the remote version. The local and remote versions have conflicting changes. The interface prompts the user to resolve the conflict. The resolved version of t1.php is shown at the bottom, with the conflict resolved by keeping the local changes and adding the remote changes.

```
<?php
echo "t1";
echo "===";
echo "---";
echo "1";

-----

<?php
echo "t1";
echo "===";
echo "1";

-----

<?php
echo "t1";
echo "===";
echo "1";
echo "dengdeng";

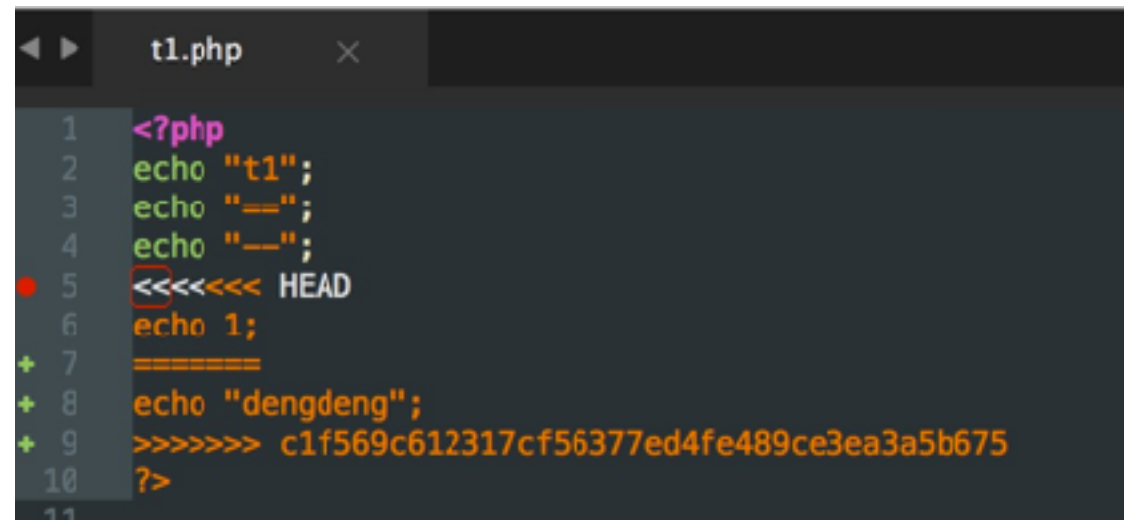
-----

?>
```

Files changed (1)

t1.php

## 2. 编辑器查看文件



The screenshot shows a code editor with the resolved t1.php file. The file contains the following code:

```
1 <?php
2 echo "t1";
3 echo "===";
4 echo "1";
5 <<<<<< HEAD
6 echo 1;
7 =====
8 echo "dengdeng";
9 >>>>>> c1f569c612317cf56377ed4fe489ce3ea3a5b675
10 ?>
```

# 常用 workflow

- 集中式 workflow
- 功能分支 workflow
- Gitflow workflow
- Forking workflow

参考: <https://github.com/xirong/my-git/blob/master/git-workflow-tutorial.md>

# 其他

- 当我们在开发新的功能的时候，需要切换到其他分支修复紧急问题，而且现在不想提交时，git stash 命令很适合处理这样的场景
  - git stash (储藏未提交已暂存的修改)
  - git stash list (列出储藏的列表)
  - git stash pop (恢复最近一次的储藏)
  - git stash apply stash@{2} (应用储藏)
- 文件忽略，忽略一些缓存、编辑器生成的、其他没必要加入版本库的文件
  - .gitignore\_global (用户目录下面的全局忽略配置文件)
  - .gitignore (项目中的忽略配置文件)
  - <https://github.com/github/gitignore> (忽略文件模板，很好、很强大)
- Sublime Text 可以安装 Git 插件
  - <https://github.com/kemayo/sublime-text-git>
- 官方资源
  - <https://git-scm.com/book/zh/v2>