

目录

2024年4月2日 16:44

[目标](#)

[Git概述](#)

[Git基本命令](#)

[Git分支操作](#)

[团队协作机制](#)

目标

2024年4月2日 11:41

Git

Git了解 分布式版本控制工具 VS 集中式版本控制工具

Git命令 熟悉Git常用命令

Git分支 分支特性、分支创建、分支转换、分支合并、代码合并冲突解决

Idea 集成Git

GitHub

创建远程仓库

代码推送 push

代码拉取 pull

代码克隆 clone

SSH免密登录

Idea集成GitHub

Gitee码云

码云创建远程仓库

Idea集成Gitee码云

码云连接GitHub 进行代码复制和迁移

Gitlab

Gitlab的搭建和部署

Idea集成Gitlab

Git概述

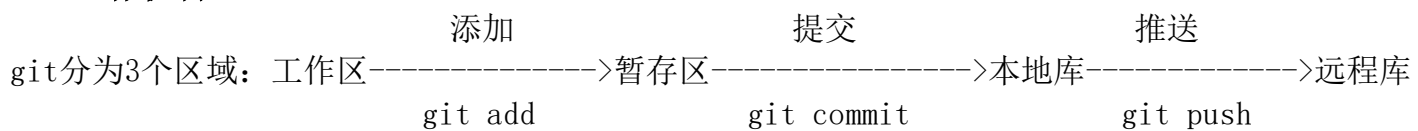
2024年4月2日 16:44

版本控制

集中式版本控制工具：CVS、SVN、VSS （中央服务器的单点故障）

分布式版本控制工具：Git（远程库）

Git 工作机制



工作区：写代码

暂存区：临时存储

本地库：历史版本（不能删除，提交本地库之前要审核）

远程库：代码托管中心（GitHub、Gitee、GitLab（局域网））

Git基本命令

2024年4月2日 16:59

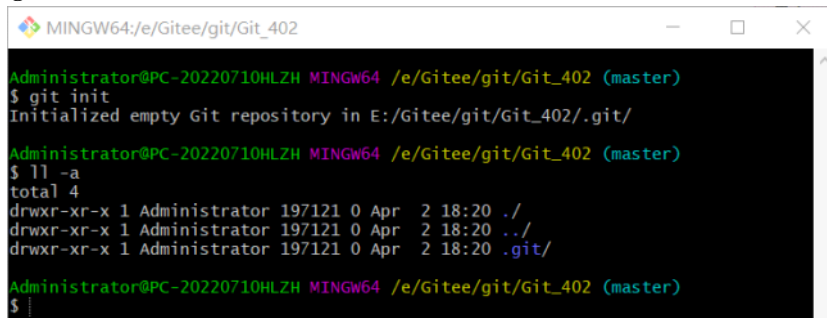
1、设置用户签名

```
git config --global user.name
git config --global user.email
name = weihongl3
email = weihong010925@163.com
```

这里设置的用户签名和其他代码托管中心没有任何关系

2、初始化本地库

```
git init
```

A terminal window titled 'MINGW64:/e/Gitee/git/Git_402' showing the execution of 'git init' and 'll -a'. The output of 'git init' is 'Initialized empty Git repository in E:/Gitee/git/Git_402/.git/'. The output of 'll -a' shows the directory structure with permissions, owner, date, and file names: './', '../', and '.git/'.

3、查看本地库状态

```
git status
```

On branch master (当前本地库分支)

No commits yet (目前没有提交、没有需要提交的内容)

nothing to commit (create/copy files and use "git add" to track)

使用vim 添加一个 hello.txt文件后再检查状态

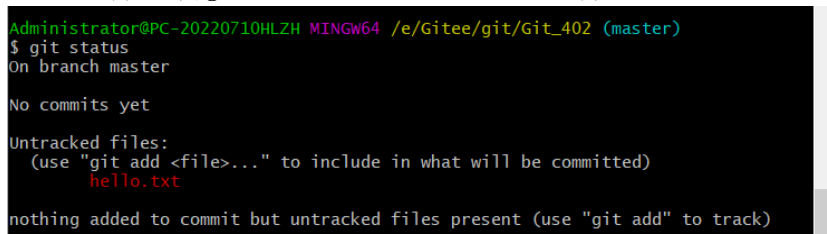
Untracked files:

(use "git add <file>..." to include in what will be committed)

hello.txt

nothing added to commit but untracked files present (use "git add" to track)

红色文件说明git目前还没有追踪到该文件

A terminal window titled 'MINGW64:/e/Gitee/git/Git_402 (master)' showing the output of 'git status'. The output indicates that there are untracked files, specifically 'hello.txt', which is highlighted in red. The message 'nothing added to commit but untracked files present (use "git add" to track)' is shown at the bottom.

3. 添加暂存区 (将工作区内容添加到暂存区)

```
git add 文件名字
```

(自动转换换行符)

warning: LF will be replaced by CRLF in hello.txt.

The file will have its original line endings in your working directory

再次查看状态（绿色文件git已经追踪到该文件，在暂存区）

```
Administrator@PC-20220710HLZH MINGW64 /e/Gitee/git/Git_402 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   hello.txt

Administrator@PC-20220710HLZH MINGW64 /e/Gitee/git/Git_402 (master)
$
```

此时文件还在暂存区，还没有上传到本地库，未形成历史版本

根据提示，可以使用(use "git rm --cached <file>..." to unstage)删除暂存区文件

4、提交本地库（提交暂存区文件到本地库）

git commit -m "日志信息" 文件名

```
Administrator@PC-20220710HLZH MINGW64 /e/Gitee/git/Git_402 (master)
$ git commit -m "提交练习" hello.txt
warning: LF will be replaced by CRLF in hello.txt.
The file will have its original line endings in your working directory
[master (root-commit) 143d619] 提交练习
1 file changed, 11 insertions(+)
create mode 100644 hello.txt
```

```
Administrator@PC-20220710HLZH MINGW64 /e/Gitee/git/
$ git status
On branch master
nothing to commit, working tree clean
```

再次查看状态，没有内容需要提交

5、查看版本信息

git reflog

版本号（精简版前七位）

提交日志

143d619 (HEAD -> master) HEAD@{0}: commit (initial): 提交练习

git log（查看详细信息）

commit 143d6196cd6b6d1e040317e261b6055ff3dcd3f3 (HEAD -> master)（完整版版本号）

Author: weihong13 <weihong010925@163.com>

Date: Tue Apr 2 18:50:46 2024 +0800

提交练习

6、修改文件（hello.txt）

（1）将hello.txt文件中的内容进行修改

（2）再次查看本地库状态

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: hello.txt

no changes added to commit (use "git add" and/or "git commit -a")

（3）再次添加、提交

git add hello.txt

git commit -m"修改后提交" hello.txt

（4）查看历史版本

git reflog

ff68116 (HEAD -> master) HEAD@{0}: commit: 修改后提交
143d619 HEAD@{1}: commit (initial): 提交练习

7、查看历史版本

git reflog (查看精简的历史版本)
git log (查看详细的历史版本)

8、版本穿梭 (去到不同版本下)

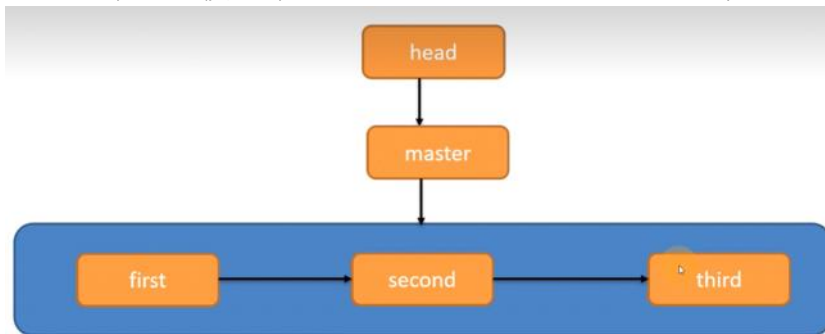
git reset --hard 版本号

git reset --hard 143d619
HEAD is now at 143d619 提交练习

通过查看日志，发现指针已经指向 提交练习的版本

```
Administrator@PC-20220710HLZH MINGW64 /e/Gitee/git/Git_402 (master)
$ git reflog
143d619 (HEAD -> master) HEAD@{0}: reset: moving to 143d619
ff68116 HEAD@{1}: commit: 修改后提交
143d619 (HEAD -> master) HEAD@{2}: commit (initial): 提交练习
```

不同版本的穿梭，本质上可以理解为该分支指针的改变



Git分支操作

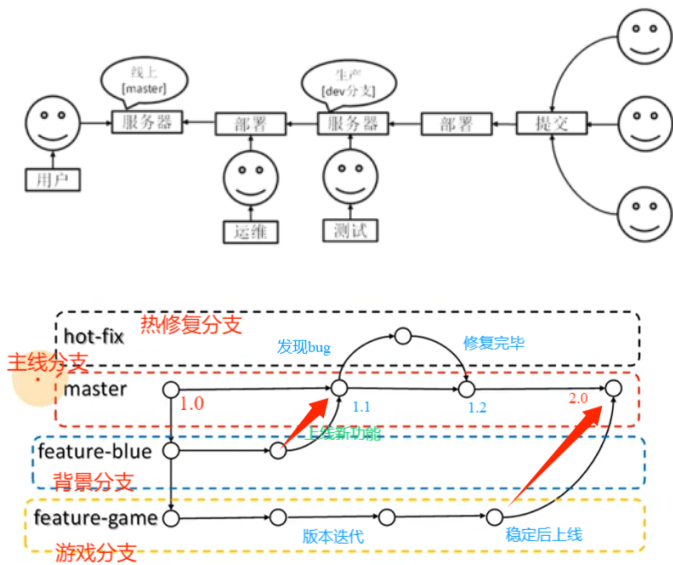
2024年4月3日 16:30

Git分支操作

Git分支操作

4.1 什么是分支

使用分支可以并行开发，提高效率，某个分支开发失败后不影响其他分支



4.2 分支操作

- 1、创建分支
`git branch 分支名`
- 2、查看分支
`git branch -v`
- 3、切换分支
`git checkout 分支名`

切换分支后，将该分支下的文件修改，另一分支的文件内容不变

4、修改分支

- 5、合并分支（正常合并）
`git merge 分支名`（将分支合并到当前分支）场景：用户分支与新版本合并

- 6、合并产生冲突
合并分支时，两个分支在**同一文件的同一位置**有两套完全不同的修改，git无法决定我们使用哪一个，必须**人为进行**决定代码谁去谁留。

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

查看状态显示

```
both modified:  hello.txt （提示两个都被修改）
分支状态变为 (master|MERGING)
```

7、解决冲突

`vim` 进入冲突文件

```
MINGW64:/e/Git
hello git!
hello weihong
hello hot-fix

<<<<<<< HEAD
master 内容
hello master merge test
=====
hello hot-fix merge test
>>>>>>> hot-fix

~
~
~
~
~
hello.txt [dos] (17:05 03/04/2024)
"hello.txt" [dos] 16L, 149B
```

将想要保留的内容留下、剩下的删除，然后再add commit（分支状态正常），此时的提交不能加文件名

```
MINGW64:/e/Git
hello git!
hello weihong
hello hot-fix

hello master merge test
hello hot-fix merge test
```

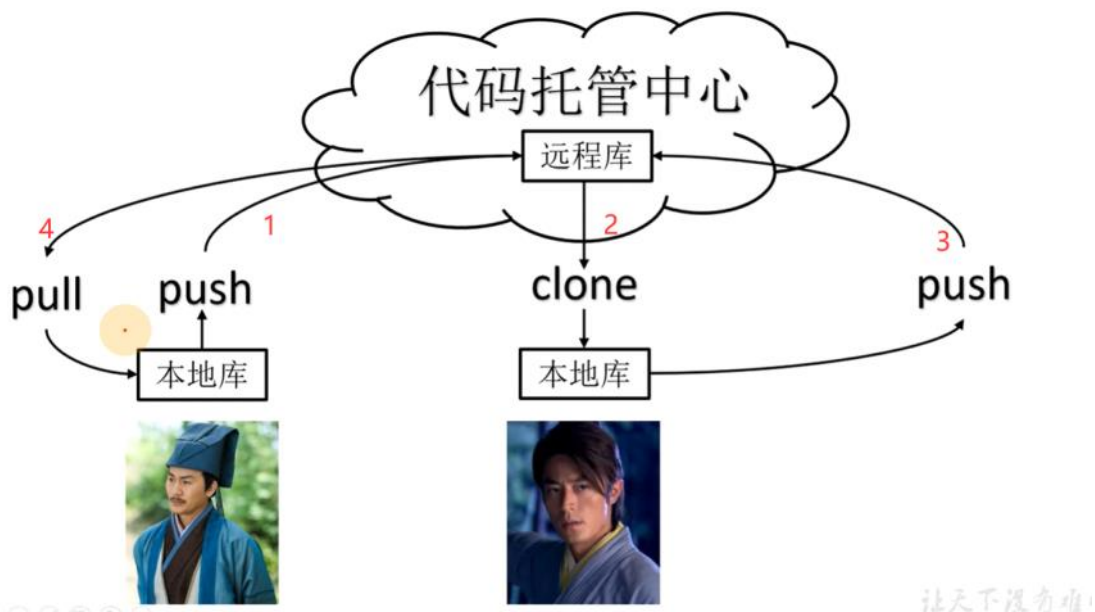

团队协作机制

2024年4月3日 17:43

团队协作机制

团队协作就涉及到代码托管中心（远程库：GitHub、Gitee等）

5.1 团队内协作



5.2 跨团队协作

