Git版本控制

git -v 显示版本

#初始配置git

git config --global user.name <Name>配置用户名（如果Name有空格则需要加上””）

git config --global user.email <Email>配置邮箱

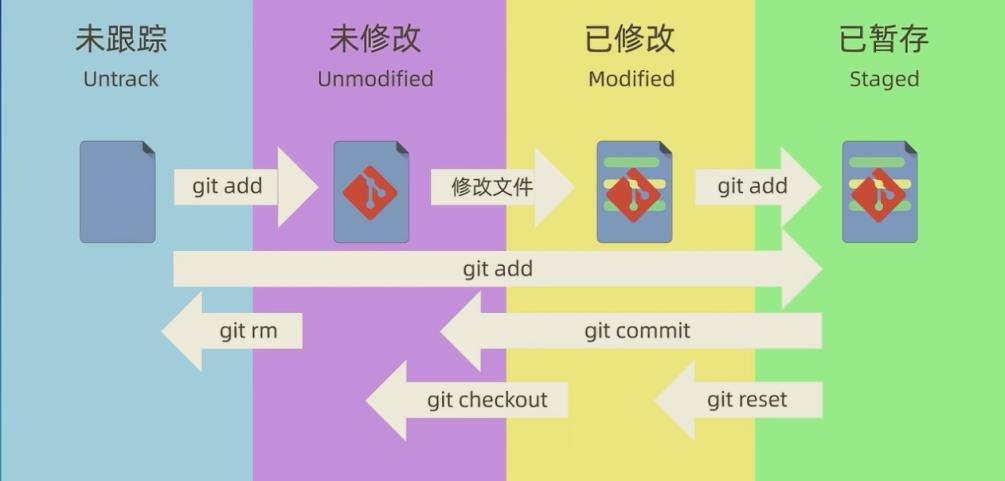
git config --global credential.help 保存用户名与密码

git config --global --list查看配置信息

上面--global代表全局配置，对所有仓库都有效

工作区域分类：工作区、暂存区、仓库区

文件状态分类：未跟踪，未修改，已修改，已暂存



git status 查看当前仓库的状态,红色的是未跟踪，绿色是已跟踪

mkdir path\<dir> 创建一个文件夹

cd <dir>切换到此文件夹

git init (<RepoName>)创建仓库

ls -a显示所有文件，如果git init之后会显示一个.git文件夹

echo .> filename.txt添加文件 echo “Hello” > filename.txt写内容

图示, 示意图

AI 生成的内容可能不正确。

git add (<FileName>)文件暂存 git add \*.cpp所有cpp文件暂存 git add **.**当前目录文件暂存

git commit -m <message>将暂存区的文件提交到仓库（不提交工作区的文件）

git stash 暂存已修改

git stash list查看所有暂存

文本

AI 生成的内容可能不正确。

git log查看提交记录 --oneline简洁 -p <file>某个文件

git log --graph --oneline --all 图形化显示分支和合并历史

git-log相当于git log --oneline

用git-log会在左边出现分支可视化图,如上图.

git reflog可以看到已经删除的提交记录

git blame <file>显示谁在什么时候修改了哪一行

git diff 查看工作区与暂存区的差异

git diff --cached 查看暂存区与最新提交的差异

git diff HEAD 查看工作区与最新提交的差异

忽略文件：创建.gitignore文件来指定Git忽略的文件：

# 忽略所有.log文件

\*.log

# 忽略特定文件

secret.txt

# 忽略文件夹

node\_modules/

build/

例：在VS2022的单个项目文件夹里面有较大的文件，无法传入GitHub，需要忽略：

# Visual Studio 临时文件

.vs/

\*.ipch

\*.suo

\*.user

bin/

obj/

图表

AI 生成的内容可能不正确。

git branch 输出所有分支

git branch -r 输出所有远程分支

git branch -v查看分支

git branch -M main指定分支名称为main

git branch -d <branch>删除分支,但是有检查

git branch -D <branch>不做检查强制删除

git branch <newBranch>创建分支，但是仍然停留在原分支

git checkout <branch>切换到分支 checkout <version>用版本号切换

git checkout -b <branch>创建并切换到分支

git merge <branch\_to\_be\_merged>将分支合并到当前分支

注：1.切换分支之前要提交或暂存修改，否则Git会阻止切换！

2.远程分支需要先拉取(fetch)才能切换。

git reset (<figure>) (<version>) 回退到特定版本，

<figure>有三个参数：

--soft保留修改，仅撤销commit,修改会保留在暂存区

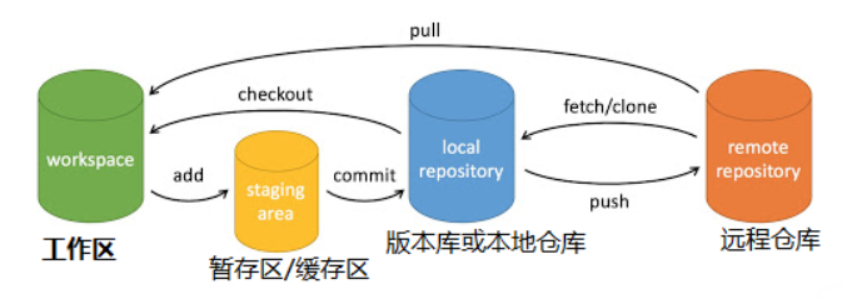
--hard彻底丢弃修改，慎用

--mixed保留修改，但撤销commit和add(修改会保留在工作区（未git add）)

<version>是版本号，如a1b2c3d，可以从git log获取。

git rm <FileName>同时在工作区和暂存区删除文件

#远程仓库：



git clone <RemoteRepoURL> (<Path>) 从远程仓库克隆，clone之后，远程仓库自动关联。

远程仓库SSH示例：git@github.com:xiao-Huahuo/FufuRepo.git

使用SSH的方式时，需要配置SSH密钥：

ssh-keygen -t rsa

最后出现一个”Enter file in which to save the key (...): ”

（1）没有配置过SSH密钥时：

输入密钥名称”test”，再次回车，(设置密码),就会在~/.ssh下生成2个id\_rsa的秘钥：

test (私钥,**保密**) ,test.pub (公钥，上传到GitHub)

打开公钥文件test.pub，复制公钥:

cat ~/.ssh/test.pub

回到GitHub，点击头像/Settings/SSH and GPG keys,

点击New SSH key,将公钥粘贴到里面。

配置密钥后，在~/.ssh创建一个config文件(**不是**config.txt**!!!**)，将以下五行加入到文件里面：

# github

Host github.com

HostName github.com

PreferredAuthentications publickey

IdentityFile ~/.ssh/test # 指定私钥路径

这个文件的意思：当访问github.com的时候指定使用test这个文件。

最后测试链接：输入

ssh -T [git@github.com](mailto:git@github.com)

如果出现”Hi xiao-Huahuo!You’ve successfullu authenticated...”之类，则认证成功。

（2）如果配置过SSH密钥时：

写密钥名test，回车，输入密码。

最后回到本地仓库**的上一级目录**，执行git clone命令，输入创建SSH密钥的时候输入的密码（如果创建的 时候没有输入密码这里也不需要输入密码，直接回车），即可成功克隆,克隆成功后将会出现一个新文件夹 (FufuRepo)。

git remote -v显示远程仓库关联状态

git remote add origin <RemoteRepoURL>将本地仓库关联到远程仓库，可给远程仓库取一个小名

（即这里的origin，下面都用origin表示远仓，main表示本仓分支）

关联远程仓库之后，在github的远程仓库里面不会立即看到本地仓库的代码，只有当推送后才会看到。

git remote remove origin 删除远程仓库

git remote set-url origin <RemoteRepoURL>修改远程仓库地址，远程仓库的小名(origin)不变

git remote get-url origin获取远程仓库的url

git diff main origin 比较本地分支和远程分支的差异

git push (--force) origin <branch> 将本地仓库同步到远程仓库,

其中<branch>为想要拉取的远程分支（如main,master）

cd FufuRepo

git add .

git commit -m “fufu仓库”

git push -u origin main #推送代码，并设定上游分支

这里面-u：将本地分支与远程分支绑定，后续只需 git push即可，无需指定分支。

--force表示强制推送，覆盖远程仓库

远程仓库有本地没有的新提交的时候，Git会拒绝push：



解决方法：

（方法一）拉取远程更改并合并（推荐）：

git pull origin main

git add .

git commit -m “Merged remote changes”

git push origin main

（方法二）强制覆盖（慎用）

git push --force origin main

远程仓库更新时的处理：

（1）git fetch origin 拉取远程仓库的更新，仅改变.git

git merge origin将远程仓库合并到本地，这里需要先fetch在merge

（2）git pull origin main 拉取并更新，相当于fetch+merge

文件较大时的处理方法:设置512MB的缓冲区,

git config --global http.postBuffer 524288000

文件极大时的处理方法:

