< 2022年3月 >

日	一	二	三	四	五	六
27	28	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

搜索

找找看

积分与排名

积分 - 444737
排名 - 1259

随笔分类 (410)

DevOps(32)
Docker(27)
Gerrit(6)
Git(11)
GitLab(5)
Infos and Data(21)
Java(10)
Jenkins(19)
Kubernetes(7)
Linux(35)
Machine Learning(24)
Maven(11)
MLCC(20)
Project(7)
Python(32)
Scala(10)
Shell(20)
SonarQube(4)
SQL(26)
Testing - Common(9)
Testing - Summary(16)
Testing - Test Automation(8)
Tools(25)
Trouble Shooting(8)
Uncategorized(13)
Visual Studio Code(4)

文章分类 (18)

反求诸己(9)
他山之玉(9)

Git - - subtree与submodule

目录

回到顶部

1 - 仓库共用(子仓库、子项目)

两种子仓库使用方式

- git submodule(子模块)
- git subtree(子树合并)

从1.5.2版本开始，官方新增Git Subtree并推荐使用这个功能来替代Git Submodule管理仓库共用(子仓库、子项目)

回到顶部

2 - submodule 与 subtree 对比

2.1 git submodule

- 可以将其他仓库某个commit作为仓库的子目录
- 克隆仓库需要额外的步骤 init 和 update
- 产生.gitmodule文件记录和submodule版本信息
- 删除submodule步骤繁琐
- 可以在子仓库单独查看子仓库的修改记录，相当与在一个单独的仓库内，对外层父仓库不可见

2.2 git subtree

- 官方推荐方式
- 不增加.gitmodule等文件
- 管理和更新流程简洁，对于项目中的其他成员透明（意味着可以不知道subtree的存在）
- 本质就是把子项目目录作为一个普通的文件目录，对于父级的主项目来说是完全透明的，原来是怎么操作现在依旧是那么操作

- 无法直接单独查看子仓库的修改记录，因为子仓库的修改包含在父仓库的记录中了。

回到顶部

3 - subtree

3.1 在父仓库中新增子仓库

```
cd <父仓库>
git subtree add --prefix=<子仓库在父仓库的相对路径> <子仓库地址> <branch> --squash

# 参数--squash: 表示不拉取历史信息，只生成一条commit信息，也就是不拉取子项目完整的历史记录
```

如果不需要更新或推送子仓库的改动，那么对于其他项目人员来说，可以不需要知道子仓库的存在。
也就是说，在这种情况下，子仓库就相当于父仓库的一个普通目录。

注意:
如果在子仓库发生改动（更新和修改）后，在父仓库中运行git status查看到子仓库文件显示modified，需要在父仓库中使用 add commit push 提交推送。
也就是说，子仓库的更改是会反映在父仓库的更改上的，因此只要是对子仓库进行了修改,无论如何都需要对父仓库进行一次提交。

示例：

```
cd temp-test-1
git subtree add --prefix=sub/temp-test-2 <temp-test-2 address> master --squash
```

此时通过git log可以查看到新增两条commit

```
commit b76c7b190760f33e7ae9dfeba40136e39309b737 (HEAD -> master)
Merge: b0aec0a e6a10bc
Author:
Date:

    Merge commit 'e6a10bc748638240ff372ae19c747584b7d8d1af' as 'sub/temp-test-2'

commit e6a10bc748638240ff372ae19c747584b7d8d1af
Author:
Date:

    Squashed 'sub/temp-test-2/' content from commit bfb58a3

    git-subtree-dir: sub/temp-test-2
    git-subtree-split: bfb58a379631813584d286ee00a19a79860f9562
```

3.2 父仓库的改动

在父仓库目录下查看状态和提交修改都和原来一样，保持不变。

3.3 拉取子仓库的更新

```
git subtree pull --prefix=sub/temp-test-2 <temp-test-2 address> master --squash
```

3.4 推送子仓库的修改

```
git subtree push --prefix=sub/temp-test-2 <temp-test-2 address> master
```

3.5 子仓库切出起点

可以将子项目当前版本切出为一个分支，作为 push 时遍历的新起点，这样以后每次遍历都只从上次切出的分支的起点开始，不会再遍历以前的了，节约时间。
这个分支只是作起点储存用的，不用管它不用修改不用推送到远程库。
需要更新这个起点时，只需要再在当前版本上再切出一个作起点的分支覆盖原来的，命令和第一次切出分支作起点时相同。

```
git subtree split [--rejoin] --prefix=<本地子项目目录> --branch <主项目中作为放置子项目的分支名>
```

注意：
如果 push 时使用了 --squash 参数合并提交，那么 split 时不能使用 --rejoin 参数，反之必须使用。

回到顶部

4- submodule

```
git clone <repository> --recursive  # 递归的方式克隆整个仓库，包含父仓库和子仓库的内容
git submodule add <repository address> <path> # 添加子仓库
git submodule init  # 初始化子仓库，向.git/config文件写入子模块的信息
git submodule update  # 更新子仓库，拉取父仓库中对应子仓库的提交id内容到到父仓库目录
git submodule foreach git pull  # 拉取所有子仓库
```

4.1 在父仓库中新增子仓库

```
cd <父仓库>
git submodule add <子仓库地址> <子仓库在父仓库的相对路径>
```

命令执行成功后

- 父仓库根目录下会产生.gitmodules文件，包含子仓库的path和url信息，并且.gitmodules在父仓库的git版本控制中
- 父仓库的git配置文件中加入了submodule字段，包含子仓库的url信息
- 父仓库.git目录下生成modudles文件夹，包含子仓库的所有相关信息

示例： 在父仓库中新增子仓库并提交子仓库信息

```
cd <project>
git submodule add <module repo addr> <module path>
git add *
git commit -m "add submodule"
git push origin master
```

4.2 拉取整个仓库

如果单纯使用git clone命令，克隆一个包含子仓库的仓库，并不会clone子仓库的内容。
需要执行本地.gitmodules初始化的命令，再同步远端submodule源码。

方式1： 获取父仓库和所有子仓库的内容

```
git clone <父仓库地址> --recursive  或者 git clone <父仓库地址> --recurse-submodules

# 使用参数--recursive，Git会自动递归去拉取所有的父仓库和子仓库的相关内容
```

方式2：

```
git clone <父仓库地址>
git submodule init && git submodule update  或者  git submodule update --init --recursive

# - git submodule init  # 初始化本地.gitmodules文件
# - git submodule update  # 同步远端submodule源码
```

4.3 修改子仓库

如果子仓库发生改动，需要先在子仓库提交，然后再到父仓库提交。
子仓库提交结束后，在父仓库的根目录执行 git status 命令会显示子仓库有新的提交。

示例：

```
cd <project>/<module>
git branch
echo "This is a submodule." > sm.txt
git add *
git commit -m "add sm.txt"
git push

cd ..
git status
git diff
git add *
git commit -m "update submodule add sm.txt"
git push
```

4.4 更新子仓库

非子仓库的开发人员只需在父仓库下pull代码时，如果发现submodule有更改，执行git submodule update进行更新，然后将改动提交到父仓库。
默认的使用git status可以看到父仓库中submodule commit id的改变。

方式1： 先pull父项目，然后执行git submodule update

```
cd <project>
git pull
git submodule update
```

方式2： 先进入子模块，然后切换到需要的分支，然后对子模块pull

```
cd <project>/<module>
git checkout master
cd ..
git submodule foreach git pull
```

4.5 删除submodule

方式1：

```
git submodule deinit -f <submodule>  # 逆初始化模块，子模块目录将被清空
git rm --cached <submodule>  # 删除.gitmodules中记录的模块信息(--cached选项清除.git/modules中的缓存)
git submodule  # 没有显示子模块信息
git commit -m "remove submodule"
```

方式2：

```
git rm -rf <子仓库在父仓库的相对路径>
rm -rf .git/modules/<子仓库名称>
vim .git/config  # 删除submodule相关的内容
git commit -m "remove submodule"
```

行动是绝望的解药！

欢迎转载和引用，但请在明显处保留原文链接和原作者信息!
本博客内容多为个人工作与学习的记录，少部分内容来自于网络并略有修改，已尽力标明原文链接和转载说明。如有冒犯，即刻删除！

以所舍，求所获，有所依，方所成。

分类: Git

好文要顶 关注我 收藏该文

Anliven
关注 - 0
粉丝 - 85
+加关注

0 推荐 0 反对

« 上一篇： Jenkins - API详解
» 下一篇： DevOps - API网关

posted @ 2020-09-16 22:37 Anliven 阅读(1461) 评论(0) 编辑 收藏 举报

刷新评论 刷新页面 返回顶部

（评论功能已被禁用）

编辑推荐：
· 神奇的 CSS，让文字智能适配背景颜色
· 戏说领域驱动设计（十五）——内核元素
· ASP.NET Core 框架探索之 Authentication
· ASP.NET Core 6框架揭秘实例演示[21]：如何承载你的后台服务
· 记一次 dump 文件分析历程

OpenHarmony开发板免费试用

最新新闻：
· 低碳水饮食短期内可能会减轻体重，但同时可能也在付出惨重代价
· 科学家发现了一桌黑洞台球，它们撑起了一个黑洞“太阳系”
· NASA延长火星直升机飞行任务到9月份，为火星车探路
· 英特尔图形学专家被AMD挖走，担任副总裁，研发实时光追技术
· 让图网络更稳健！谷歌提出SR-GNN，无惧数据标记偏差和领域转移
» 更多新闻...

Copyright © 2022 Anliven
Powered by .NET 6 on Kubernetes