链滴

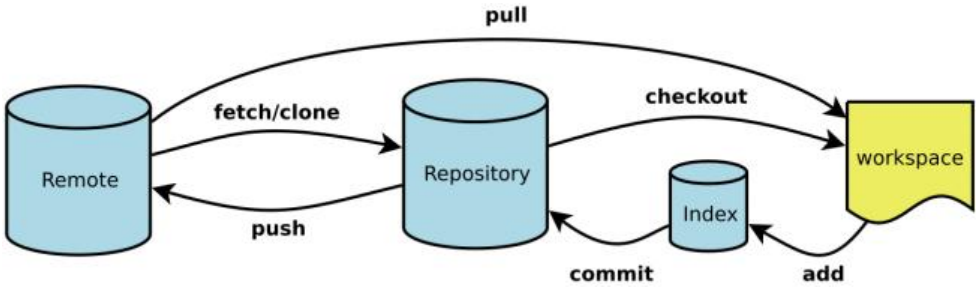
Git 常用命令

作者：yangin

原文链接：https://ld246.com/article/1574405797661

来源网站：链滴

第一部分：Git 原理图



Workspace：工作区

Index / Stage：缓存区

Repository：本地仓库

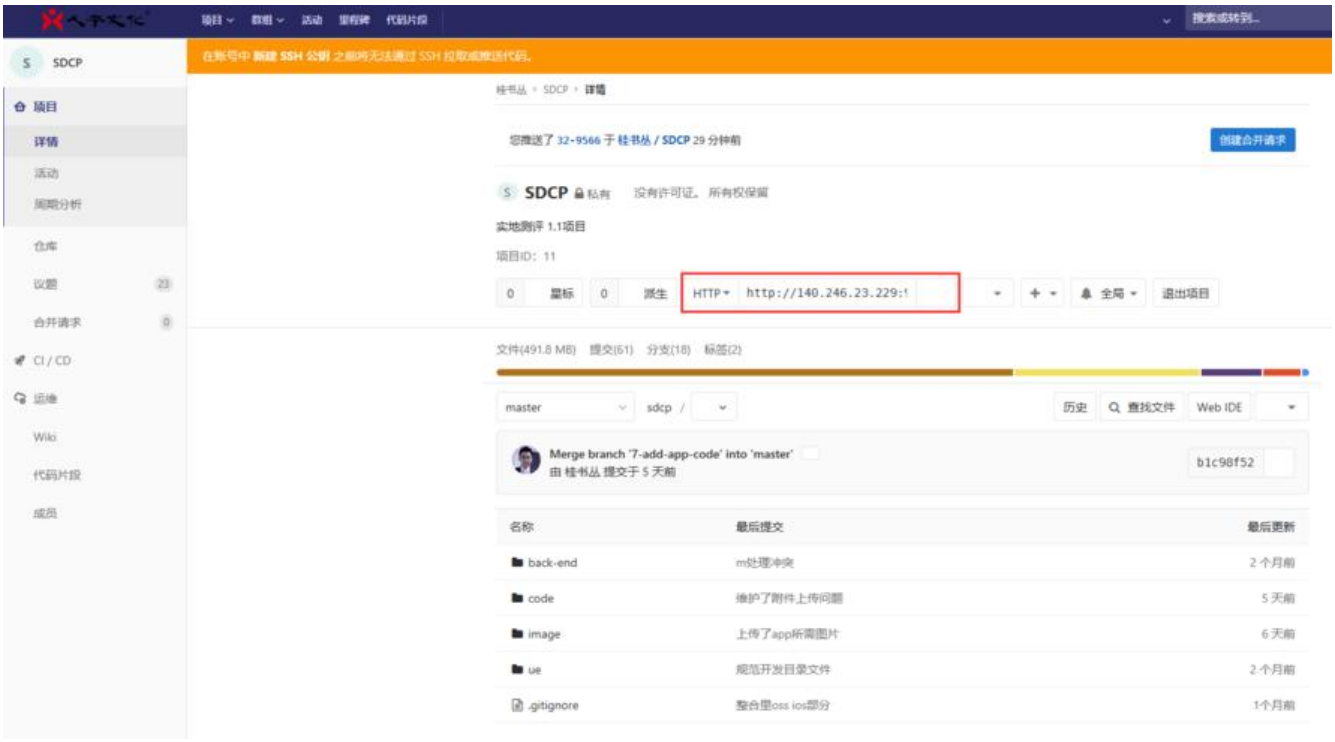
Remote：远程仓库

第二部分：新建分支处理问题步骤

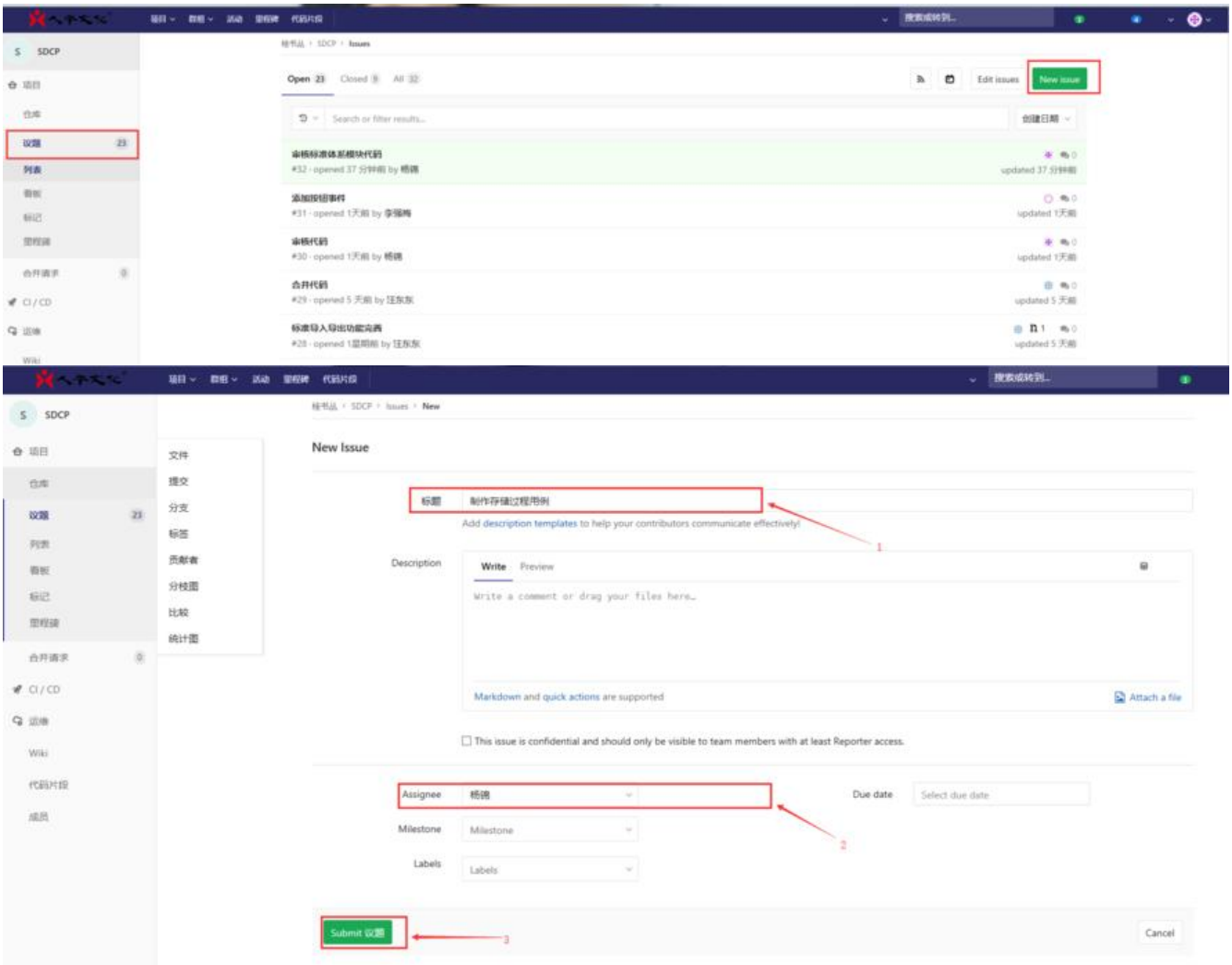
第一步：在本地新建文件夹，并拉取 Git 目标仓库代码。

Git clone [仓库地址]

说明：仓库地址一般在如图位置，为 http 地址。

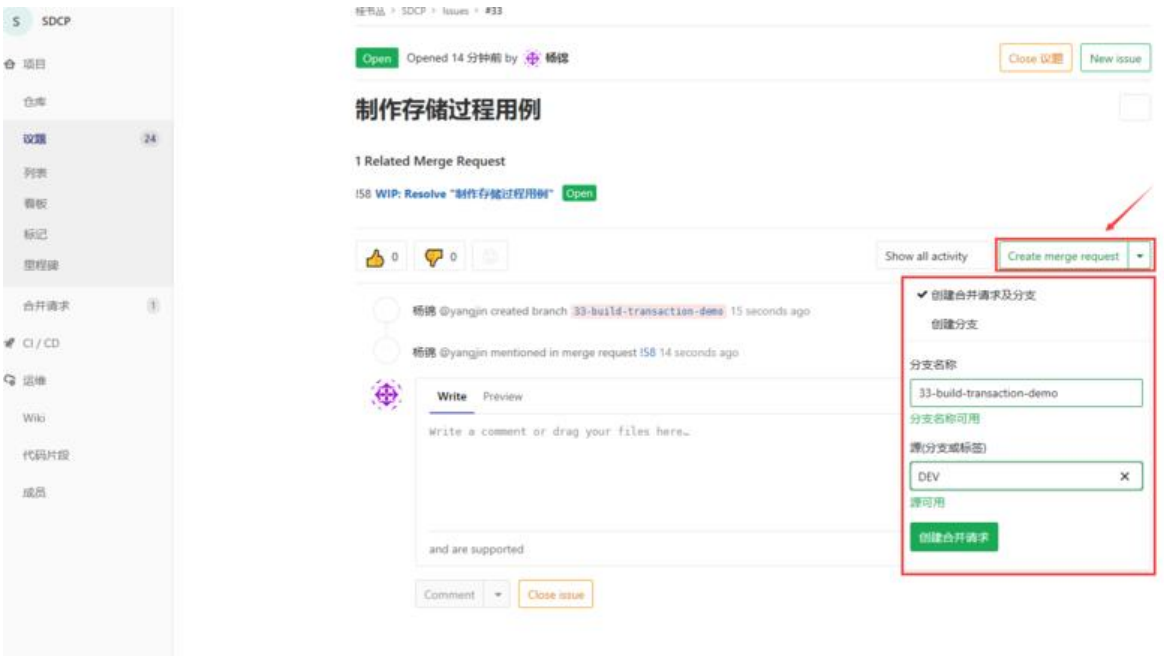


第二步：在 GitLab 上新建议题。



第三步：在新建议题上新建分支。注意选择分支的来源。

原文链接：Git 常用命令



第四步：在本地终端，执行 pull 命令，获取最新的分支信息。

Git pull

第五步：查看远程分支列表，并确定目标分支名称。

Git branch -r

确定目标分支：33-build-transaction-demo

第六步：检出目标分支到本地，并切换到工作区

Git checkout -b 33-build-transaction-demo origin/33-build-transaction-demo

第七步：在本地运行项目，便可进行代码编辑。

第三部分：切换工作区分支

第一步：检查本地分支

```
Git branch      //查看本地分支
Git branch -r   //查看远程分支
Git branch -a   //查看所有分支（包括本地与远程）
```

第二步：切换分支

Git checkout [分支名]

注意：当工作区有文件发生修改时，切换分支，需要先将原修改部分提交到本地仓库，或者还原到修前。

第四部分：提交代码步骤

第一步：检查发生修改的文件

原文链接：Git 常用命令

```
Git status
```

第二步：添加修改文件到缓存区

```
Git add .                    //添加所有修改文件
Git add [file1] [file2] ...  //添加指定文件
Git add [dir]                //添加指定目录下修改的文件
```

说明：以上方式 3 选 1。

第三步：提交缓存区文件到本地仓库。

```
Git commit -m '提交描述'
```

第四步：在分支下推送本地仓库到远程 Git 仓库。

```
Git push
```

第五步：至此，完成了本地代码的提交。此分支的作用也就完成了，可以删除本地分支了。

```
Git branch -d [删除分支名]
```

注意：要删除分支，需要先将工作区切换到其他分支，然后再删除目标分支。

第五部分：审核代码步骤

1.检查分支代码问题

第一步：确定服务器运行代码分支，即主版本分支，如 dev。

第二步：本地检出远程 dev 分支,并切换到该分支

```
Git checkout -b dev origin/dev
```

说明：在本地新建一个与远程相同名称的分支，并关联，注意，本地分支名与远程分支名必须相同。

第三步：此时，工作区的代码为本地 dev 分支的代码，跑起来，确认没问题。

第四步：将远程目标分支检出到本地，并切换到工作区。如远程目标分支名为 add_user_module。

```
Git checkout -b add_user_module origin/add_user_module
```

第五步：运行 add_user_module，检查其是否能正常运行。

第六步：查看本地分支情况

```
Git branch
```

说明：因为前期做了两次 checkout，加上 master，此时本地应该有 3 个分支，分别为 master、de
、add_user_module

第七步：切换到主版本分支 dev

```
Git checkout dev
```

第八步：合并目标分支到当前分支（主版本分支）

```
Git merge add_user_module
```

第九步：在终端查看合并过程中的冲突问题。

第十步：运行合并后的代码，检查其是否报错。

第十一步：通过可视化界面，对比合并后的代码，add_user_module 分支相对于 dev 做的改变。

```
gitk
```

说明：执行此命令后会调出 GUI 图形界面，审核变化的代码。

2.添加审批备注

第一步：新建议题，以待审批的分支作为分支源，建立新的分支，格式可以是：[源分支名]-fix。

第二步：检出进行审批操作的分支。

```
Git checkout -b add_user_module-fix origin/add_user_module-fix
```

第三步：运行代码，在该分支上进行修改，插入审批内容。

第四步：审批完后，提交当前审批分支到远程仓库。

第六部分：撤回操作

1.撤回 Git add

```
Git reset head      //撤回所有 Git add 的数据
Git reset head [file1]  // 撤回指定 Git add 的文件
```

2.撤回 Git commit

```
Git commit --amend -m '提交描述'
```

原理：采用的是重新提交，并覆盖最后一次 Git commit 的结果。

3.撤回对文件的修改，未执行 Git add 操作

```
Git checkout [ 文件名]  //撤销指定文件的修改
Git checkout .        //撤回对所有文件的修改
```

4.显示提交的 log

```
Git log -3
```

结果如下：

```
$ git log -3
commit 4dc08bb8996a6ee02f
Author: Mark <xxx@xx.com>
Date:   Wed Sep 7 08:08:53 2016 +0800

    xxxxx

commit 9cac9ba76574da2167
Author: xxx<xx@qq.com>
Date:   Tue Sep 6 22:18:59 2016 +0800

    improved the requst

commit e377f60e28c8b84158
Author: xxx<xxx@qq.com>
Date:   Tue Sep 6 14:42:44 2016 +0800

    changed the password from empty to max123
```

5.撤回到指定的版本

Git reset --hard [提交编号]

如：

Git reset --hard e377f60e28c8b84158

第七部分：其他

1.扩充 Git 缓存区大小

Git config http.postBuffer 524288000

说明：解决 Git clone 下载过程中下载文件中断问题，可能因下载的文件过大，超出了 Git 的 postBuffer 值。此处将缓存值设置为 500M。

2.中文目录乱码处理

Git config --global core.quotepath false

3.gitk 中文图形界面乱码处理

Git config --global gui.encoding utf-8