版本控制系统

VCS：Version Control System

作用：

1. 备份代码，保存各个版本代码的修改
2. 恢复到之前的版本
3. 查看不同版本代码之间的差异
4. 支持团队协作开发，多个人可以同时修改同一份代码。

安装Git

Linux：

```
sudo apt-get install git
```

Windows：

从官方网站 https://git-scm.com 下载Windows的安装包，双击安装。

使用Git

```
git <子命令> [参数1] [参数2] ...
```

获取帮助

```
git <子命令> --help
```

配置用户名和电子邮箱

git在提交代码之前必须配置用户名和电子邮箱，不配置的话不能使用

```
git config --global user.name "LIU Yu"
git config --global user.email "liuy_xa@hqyj.com"
```

注意系统，全局和本地配置的区别。

- 系统配置（--system）　　/etc/gitconfig　对系统中所有用户的所有代码仓库生效
- 全局配置（--global）　　$HOME/.gitconfig 对当前用户的所有代码仓库生效
- 本地配置（--local）　　工作目录/.git/config 只对当前的代码仓库生效

配置优先级：local > global > system

查看配置

```
git config --list #列出所有配置
git config --global --list #只显示全局配置
```

删除配置

```
git config --global --unset core.editor
```

编辑器

git在提交代码的时候会调用文本编辑器让用户编辑提交信息，默认使用的是GNU Nano编辑器。

Git基本概念

1. 工作目录（工作区）（Working directory）：存放代码的目录
2. 代码（仓）库（Repository）：存放所有版本代码的数据库
3. 提交（commit）：把工作目录中的代码保存到代码库中
4. 检出（checkout）：把代码从代码库中恢复到工作目录中。

以上概念在大部分的版本控制系统中都有

5. 暂存区（Staging Area）：类似于购物车，用来暂存一次修改涉及到的多个文件。通常一次提交只涉及一个特性（功能）或一个bug修改，如果涉及多个文件，需要将这些文件先加到暂存区，然后统一提交到代码库。方便查看版本间的差异和回退修改。Git特有的概念。

保存修改

1. 创建工作目录

```
mkdir work #目录名自己定
```

2. 创建本地代码库

git的本地代码库一般放在工作目录中，方便管理。

```
cd work
git init
```

如果git输出`Initialized empty Git repository in /home/student/work/.git/`表示代码库创建成功。

查看工作区状态

在工作目录下执行：

```
git status
```

Untracked file（未跟踪文件）：没有放到代码库中管理的文件

向暂存区中增加文件或修改

```
git add 文件名
```

提交修改到代码库

```
git commit
```

git在提交时会调用文本编辑器，用户需要在编辑器中编写提交日志。如果提交日志为空，git不会提交修改到代码库，输出`Aborting commit due to empty commit message.`。

git使用的文本编辑器可以通过修改core.editor进行配置。

git commit使用`-m`参数时，直接将命令行中的信息作为提交日志，不再调用文本编辑器。

git commit使用`-a`参数时，直接将**已跟踪**文件的修改提交到代码库。

查看提交历史

```
git log

commit 874f2513cd966ac6643afecc36a39697969de5e9 (HEAD -> master)
Author: LIU Yu <liuy_xa@hqyj.com>
Date:   Thu Mar 26 14:24:26 2020 +0800

    add readme.txt
```

提交ID（874f2513cd966ac6643afecc36a39697969de5e9）：40个字符的随机值，唯一标识一次提交，可以使用前7个字符代替。

提交人（Author）：可以通过user.name和user.email进行配置。

提交时间

提交日志

可以使用`--oneline`显示简短的提交历史信息

```
git log --oneline
874f251 (HEAD -> master) add readme.txt
bb950e4 add hello git
ed38f47 add return value
2e62fbf create test file
```

比较两个版本之间的差异

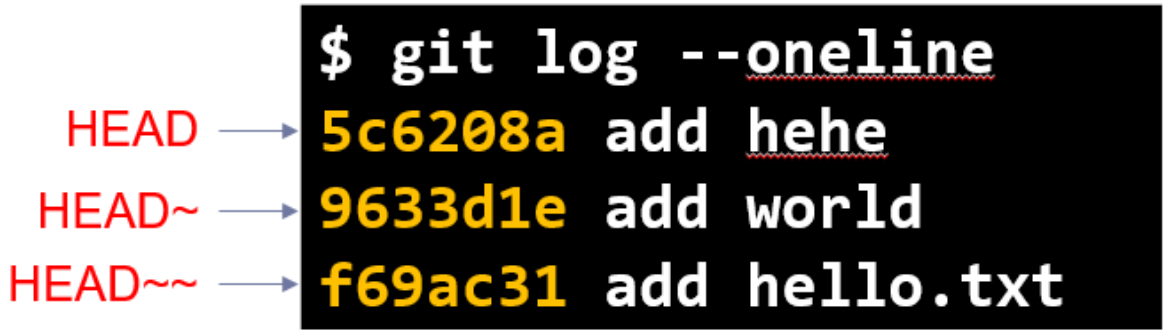
比较工作目录中的内容和代码库中最新版本之间的差异

```
git diff
```

比较任意两个版本之间的差异

```
git diff commit_id1 commit_id2
```

可以使用HEAD代替提交ID，HEAD可以看作指向当前分支的最新版本的别名。



恢复文件

将代码库中的最新版本检出到工作目录

```
git checkout HEAD .
```

HEAD代表当前分支的最新版本，可以换成提交ID，句点用来通配所有文件，也可以只回退单个文件。

重命名

```
git mv 原文件名 新文件名
git commit
```

注意：工作目录中的所有修改，只有在提交后才能保存到代码库。

删除文件

```
git rm 文件名
git commit
```

只是从最新版本中删除此文件，以前的版本中的文件仍然保存在代码库中。

使用`--cached`参数，仅删除代码库中的文件，不删除工作目录中的文件，即将已跟踪的文件变为未跟踪。

打标签

给提交ID取个别名，后续在使用提交ID的命令中，可以用标签代替提交ID。

```
git tag 标签名
```

列出所有标签

```
git tag
```

查看标签和提交的对应关系

```
git log --decorate=short
```

给以前的版本打标签

```
git tag 标签名 提交ID
```

打包代码

git会根据打包文件的扩展名决定打包文件的格式。

```
git archive 提交ID/HEAD/标签 --prefix=前缀 -o 打包文件名.扩展名
```

注意：前缀如果以`/`结尾，则将代码放到以前缀命名的目录中。

分支

分支就是代码库的一个副本。

创建分支

```
git branch 分支名
```

查看分支

```
git branch
```

分支名是绿色并且有*号标识的，是工作目录的当前分支。

切换当前分支

```
git checkout 分支名
```

切换分支成功后，Git会显示`Switched to branch '分支名'`。

在不同分支上所提交的修改，只保存在各自分支内。工作目录中的内容是当前分支的最新版本。

学习git分支操作：https://learngitbranching.js.org

分支合并

冲突（conflict）：在两个分支中分别修改同一个文件中的同一行，在分支合并时会发生冲突，Git不能自动完成合并，需要手工解决冲突。

```
Auto-merging test.c
CONFLICT (content): Merge conflict in test.c
Automatic merge failed; fix conflicts and then commit the result.
```

解决冲突：编辑发生冲突的文件，根据具体的业务逻辑修改代码，并删除“<>=”开始的标记行，重新提交代码。

当两个分支的修改内容没有发生冲突的时候，可以自动合并。

```
Auto-merging test.c
Merge made by the 'recursive' strategy.
 test.c | 7 +++++++
 1 file changed, 7 insertions(+)
```

撤销合并操作

```
git merge --abort
```

删除分支

```
git branch -d 分支名
```

删除成功后输出`Deleted branch 分支名.`

撤销提交

```
git reset 提交ID
```

将HEAD指针指向前面的提交，此提交之后的操作就撤销了。

找回之前撤销的提交

```
git reflog #找到之前撤销的提交ID
git reset 提交ID  #将HEAD指针指向之前撤销的提交
```

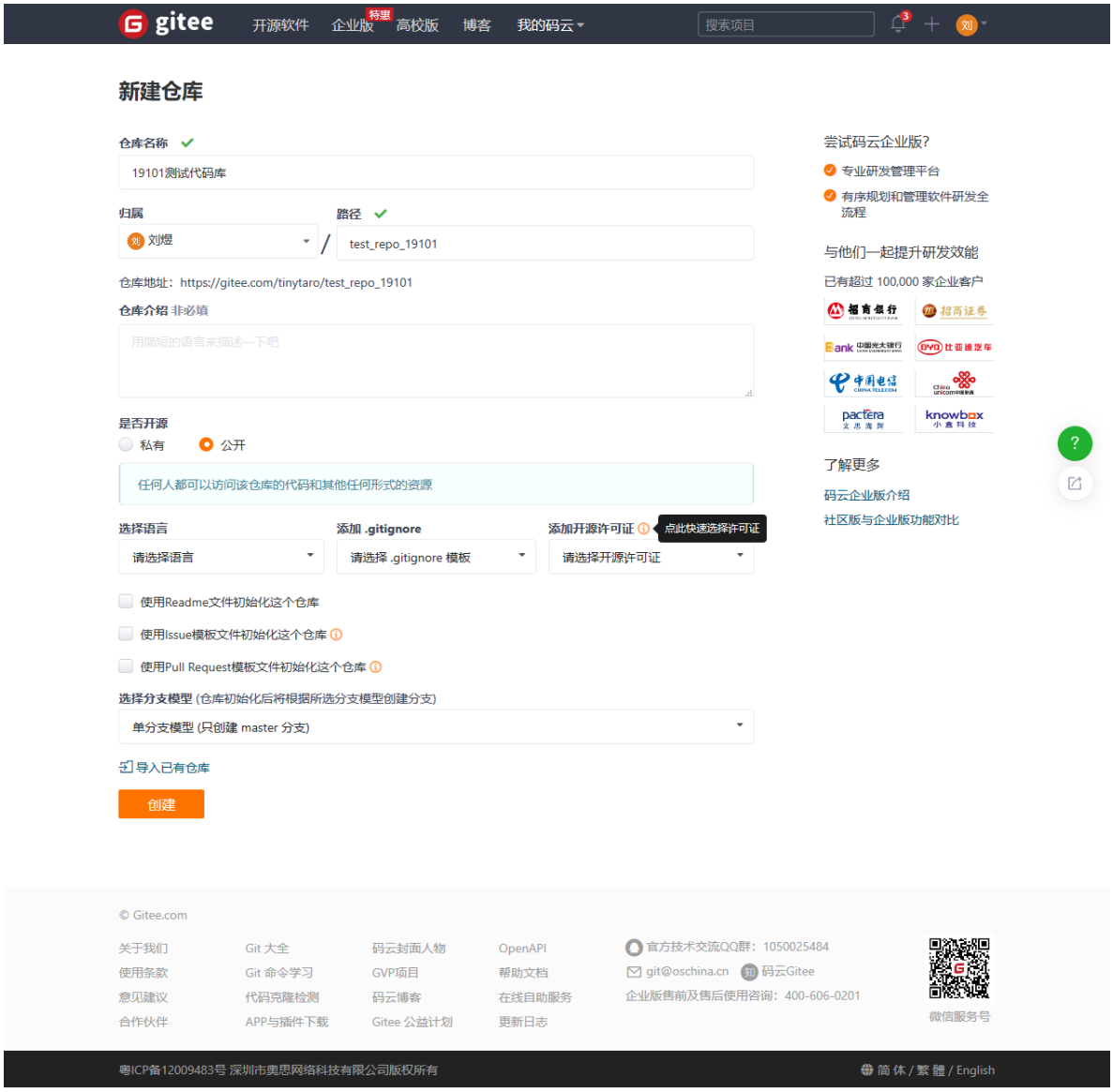
远程代码库

github/码云提供远程代码库托管服务，还有社交网站的部分功能。

创建远程代码库

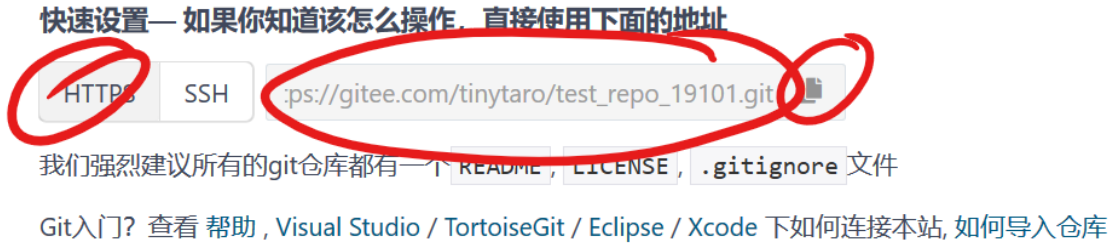
1. 

2.



关联本地代码库和远程代码库

1. 复制远程代码库地址



2. 为本地代码库添加远程代码库

```
git remote add origin https://gitee.com/tinytaro/test_repo_19101.git
```

远程代码库名字一般使用`origin`。

3. 查看关联的远程代码库

```
git remote -v
```

git会显示远程代码库的名字和地址，地址有两个，分别对应上传（push）和下载（fetch）的远程代码库地址。

同步本地代码库到远程

第一次同步时需要设置默认的远程分支

```
git push -u origin master
```

后续同步只需要执行

```
git push
```

需要输入托管网站注册的用户名和密码。

同步成功后输出：

```
$ git push
Username for 'https://gitee.com': tinytaro
Password for 'https://tinytaro@gitee.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 670 bytes | 670.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: Powered by GITEE.COM [GNK-3.8]
To https://gitee.com/tinytaro/test_repo_19101.git
   d067f8c..d8afe56  master -> master
```

下载远程代码库

1. 复制远程代码库地址

2. 执行 `git clone` 命令下载远程代码库，此命令会自动创建代码库同名的目录作为工作目录。

```
git clone https://gitee.com/tinytaro/test_repo_19101.git
```

同步远程代码库到本地

```
git pull
```

执行成功后输出:

```
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://gitee.com/tinytaro/test_repo_19101
   d8afe56..0920053  master     -> origin/master
Updating d8afe56..0920053
Fast-forward
 readme.md | 46 ++++++++++++++++++++++------------------------
 1 file changed, 22 insertions(+), 24 deletions(-)
```

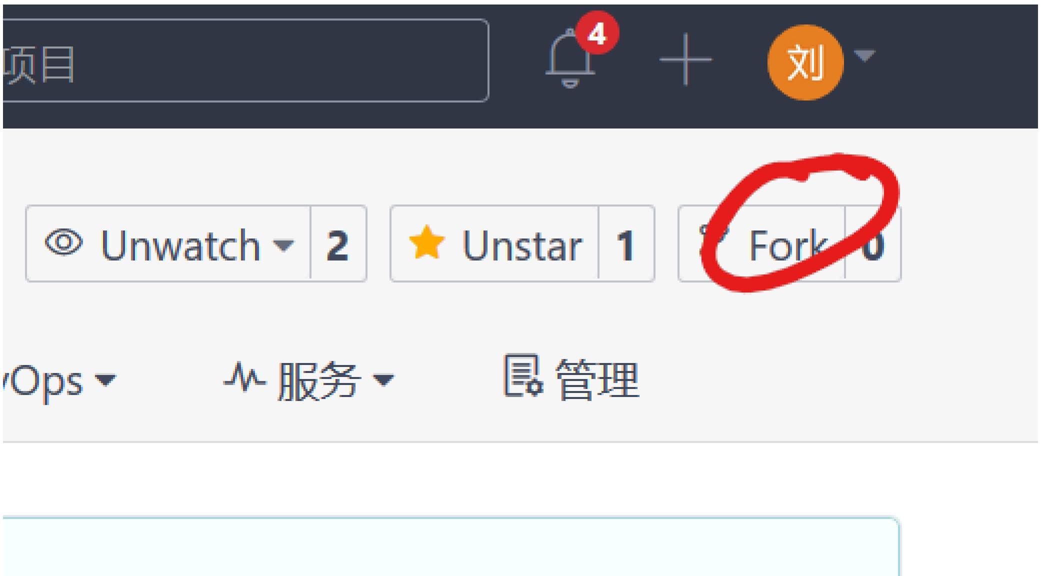
团队开发

集中式开发流程

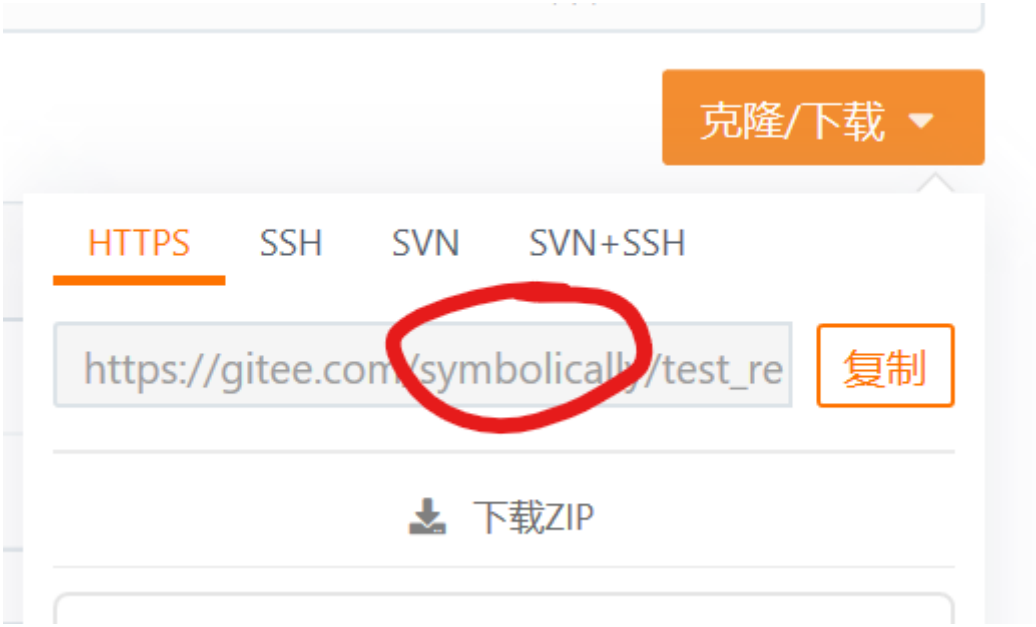
将团队的所有成员添加到项目中，每一个人都有修改代码库的权限。

集成管理员开发流程

1. 点击fork复制主代码库到自己账号



2. 下载**副本**代码库到本地




```
git clone https://gitee.com/symbolically/test_repo_19101.git
```

注意：要下载自己fork复制之后的副本代码库，不是主代码库。

3. 修改工作目录中的代码，并提交到远程副本库。

```
git add yanjianwei.txt
git commit -m "xxxxx"
git push
```

4. 创建PR（Pull Request）

Pull Request：请求配置管理员将自己修改的代码拉取到主代码库中，即将自己的修改合并到主代码库中。

需要在PR的编辑界面，写清楚PR名称（修改的内容），PR的描述，点击创建。

5. 配置管理员收到PR后，将PR的修改合并到主代码库。
6. 将主代码库和本地代码库进行关联

```
git remote add upstream https://gitee.com/tinytaro/test_repo_19101.git
```

7. 从主代码库拉取代码，更新本地代码库

```
git pull upstream master
```

8. 将本地代码库的更新推送到副本库中

```
git push
```

注意事项

- 不要提交未完成的代码（编译和测试通过，避免持续集成失败）
- 一次提交不要涉及多个特性或问题
- 避免提交二进制文件和临时文件
- 提交信息中需要说明修改原因
- 目录和文件名尽量不要使用中文，避免在不同系统上出现乱码。

作业

复制“19101测试代码库”，增加以自己名字拼音命名的txt文件，提交PR到主代码库。