

基于阿里云云效Codeup的Git代码管理

实验环境

Ubuntu 20版本

实验内容及步骤

构建实验环境

准备本地开发环境

在本地环境安装Git和python3

在终端输入指令安装Git并检查是否安装成功：

```
xinyu-yaoer2@ubuntu:~/Desktop$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 136 not upgraded.
Need to get 0 B/5,525 kB of archives.
After this operation, 38.8 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Selecting previously unselected package liberror-perl.
```

```
Processing triggers for man-db (2.9.1-1) ...
xinyu-yaoer2@ubuntu:~/Desktop$ git --version
git version 2.25.1
xinyu-yaoer2@ubuntu:~/Desktop$
```

配置全局用户名与邮箱。为标识身份，在使用 Git 前，建议执行两个全局配置，即用户名与电子邮箱。当使用 Git 提交代码时，用户名和电子邮箱将被同时记录，用以标识提交者信息，推送时亦然。以本地环境Ubuntu为例，若未配置 Git 用户名与电子邮箱，提交代码后Git 记录的提交者为当前 Ubuntu 本地用户名。

```
git config --global user.name "<username>"
git config --global user.email "<email>"
```

查看配置情况，输入以下指令：

```
git config -global user.name
git config --global user.email
```

```
xinyu-yaoer2@ubuntu:~/Desktop$ git config --global user.name "xinyuyaoer"
xinyu-yaoer2@ubuntu:~/Desktop$ git config --global user.email "xinyuyaoer@gmail.com"
xinyu-yaoer2@ubuntu:~/Desktop$ git config --global user.name
xinyuyaoer
xinyu-yaoer2@ubuntu:~/Desktop$ git config --global user.email
xinyuyaoer@gmail.com
```

随后安装Python3和Python虚拟环境，输入以下指令：

```
sudo apt install python3
sudo apt install python3-venv
```

```
xinyu-yaoer2@ubuntu:~/Desktop$ sudo apt install python3
[sudo] password for xinyu-yaoer2:
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3 is already the newest version (3.8.2-0ubuntu2).
python3 set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 81 not upgraded.
xinyu-yaoer2@ubuntu:~/Desktop$ sudo apt install python3-venv
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  python-pip-whl python3-distutils python3-lib2to3 python3.8-venv
The following NEW packages will be installed:
  python-pip-whl python3-distutils python3-lib2to3 python3-venv python3.8-venv
0 upgraded, 5 newly installed, 0 to remove and 81 not upgraded.
Need to get 2,029 kB of archives.
After this operation, 4,465 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 python-pi
```

接着查看Python版本，输入以下指令：

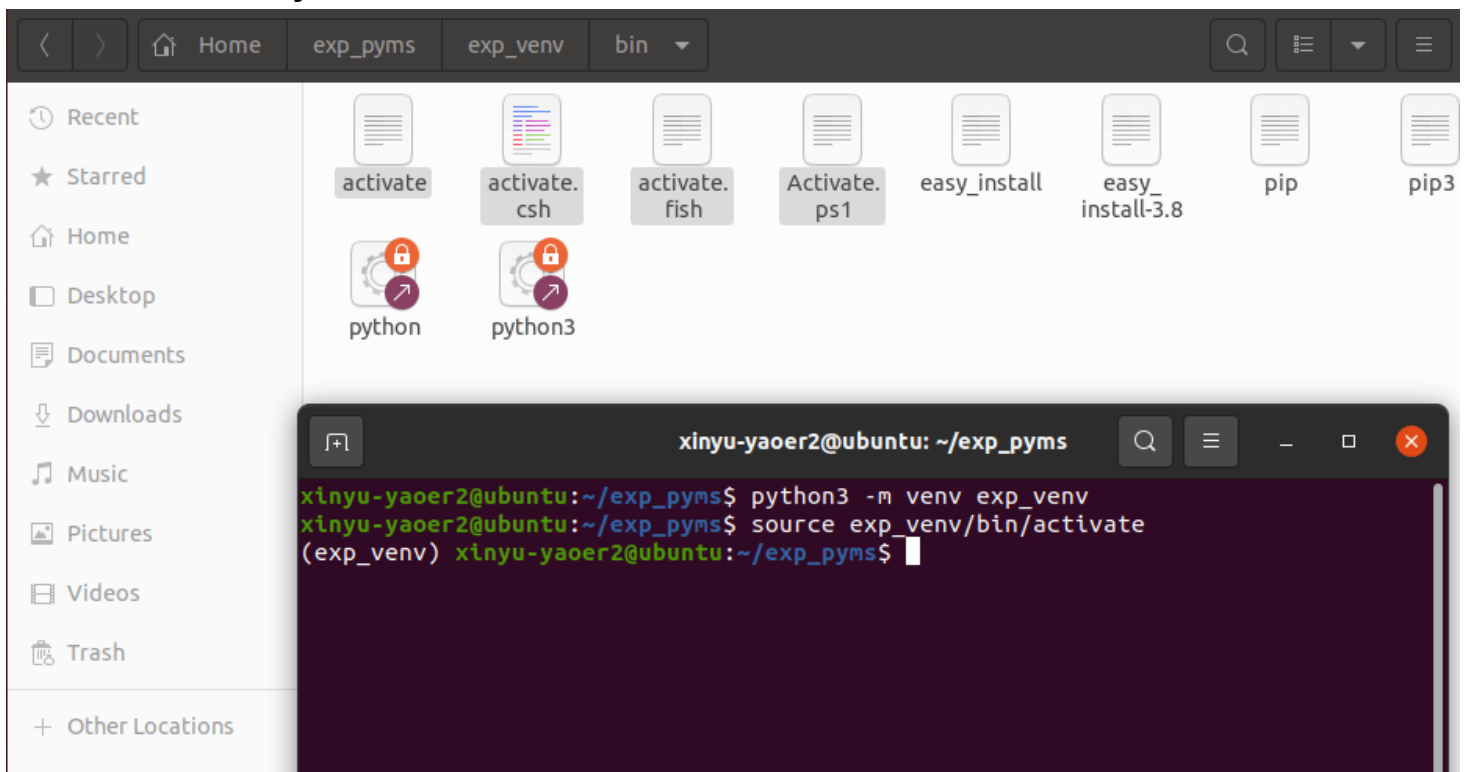
```
python --version
```

```
xinyu-yaoer2@ubuntu:~/Desktop$ python3 --version
Python 3.8.10
xinyu-yaoer2@ubuntu:~/Desktop$
```

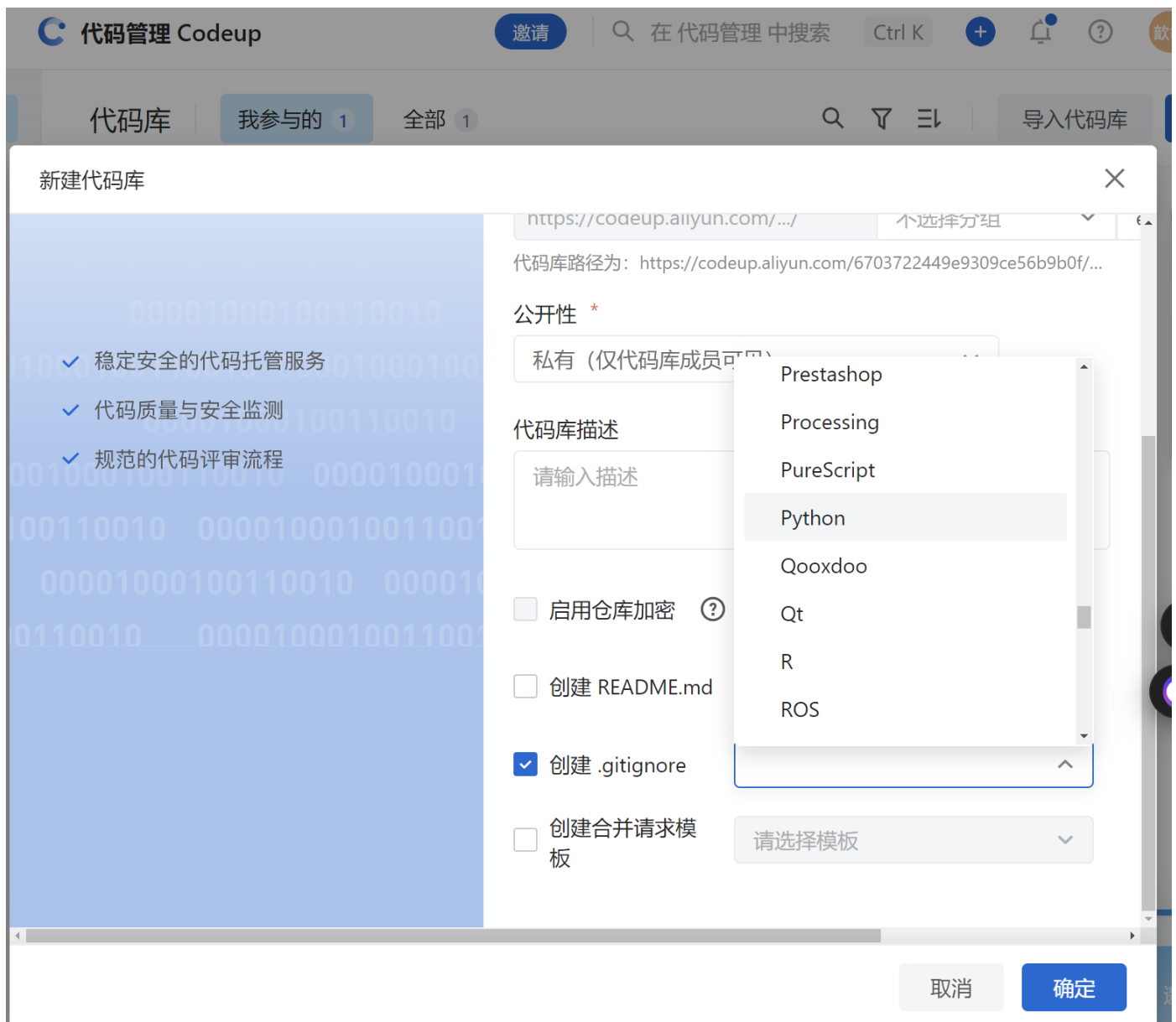
在本地安装Python第三方库

进入本地工作目录 `exp_pyms`，**创建 Python 虚拟环境** `pyms_venv`，并使用 **source 命令** 激活虚拟环境。

Python 虚拟环境可以实现为不同的项目设置独立的依赖库，以实现在统一系统中不同项目所需或者一些软件包的隔离。例如开发环境中的两个项目都依赖同一个软件包，但依赖于不同的版本，依赖是相互冲突的，那么使用 **Python 虚拟环境** 就能为不同的项目创建独立的虚拟环境。



然后配置 `requirement.txt` 文件（需要先手动创建文件），在里面编写要安装的Python库名，如本系列实验后续会使用到的web服务器框架 `sanic`、http基本库 `requests` 以及用于访问数据库的基础库 `pymysql`，内容如下：



克隆远程仓库至本地仓库

一般支持HTTPS和SSH两种协议。此次首先使用HTTPS协议。

1. 配置HTTPS克隆密码

进入Codeup系统，右上角选择个人设置，随后进入[HTTPS 密码] 页面配置克隆密码

2. 复制代码库地址 (HTTPS URL)

云端进入代码库 exp_pyms 首页，找到代码库地址，选择HTTPS，复制代码库的URL，用于仓库地址，记为 repo_addr。

3. 克隆远程仓库

本地进入用于存放工作目录的主目录，使用命令git clone 克隆远程仓库。需查看个人设置中的用户名和密码，并在本地终端正确输入远程仓库的用户名和密码，以完成身份验证。

```
(exp_venv) xinyu-yaoer2@ubuntu:~/exp_pyms$ git clone https://codeup.aliyun.com/6703722449e9309ce56b9b0f/exp_pyms.git
Cloning into 'exp_pyms'...
Username for 'https://codeup.aliyun.com': 87529617864492384
Password for 'https://87529617864492384@codeup.aliyun.com':
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 1.30 KiB | 133.00 KiB/s, done.
(exp_venv) xinyu-yaoer2@ubuntu:~/exp_pyms$
```

4. 验证

在本地终端输入指令，查看是否克隆成功。

克隆操作会在本地生成工作目录 exp_pyms，其中包含隐藏目录.git，该目录内容为 Git 仓库核心配置，不可轻易修改，同时也包括云端新建代码库是生成的.gitignore 文件，已被克隆到本地。工作目录exp_pyms 同时还包括远程仓库主分支Master中全部文件的工作副本(WorkingCopy)，当时此时有.gitignore 文件。

操作如图：

```
Unpacking objects: 100% (3/3), 1.30 KiB | 133.00 KiB/s, done.
(exp_venv) xinyu-yaoer2@ubuntu:~/exp_pyms$ ls
exp_pyms  exp_venv  requirements.txt
(exp_venv) xinyu-yaoer2@ubuntu:~/exp_pyms$ cd exp_pyms/
```

```
(exp_venv) xinyu-yaoer2@ubuntu:~/exp_pyms/exp_pyms$ ls -a
.  ..  .git  .gitignore
(exp_venv) xinyu-yaoer2@ubuntu:~/exp_pyms/exp_pyms$
```

配置本地仓库

按需配置添加 .gitignore、Dockerfile 等配置文件以及本地代码文件。本实验中，.gitignore 文件在创建远程仓库时已根据模版生成，并同步到本地，无需再单独添加。此时以 README.md 文件为例说明如何添加文件到本地仓库。

1. 在本地工作目录 exp_pyms 中添加 README.md 文件，内容为实验内容。
2. 执行 git add. 命令，将本地工作目录中所有文件添加到暂存区。

具体操作如图：


```
(exp_venv) xinyu-yaoer2@ubuntu:~/exp_pyms/exp_pyms$ touch README.md
(exp_venv) xinyu-yaoer2@ubuntu:~/exp_pyms/exp_pyms$ git add README.md
(exp_venv) xinyu-yaoer2@ubuntu:~/exp_pyms/exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   README.md
```

同步本地仓库至远程仓库

执行以下操作，完成后查看远程仓库，已出现本次提交的 [README.md](#) 文件

提交前应养成习惯，先从远程仓库pull最新代码到本地进行冲突检查，当然此时我们这个使用没有冲突。

```
git pull # 先pull
```

将全部暂存文件提交至本地仓库，当前只有README.md文件

```
git commit -m "docs: 创建README.md文件"
```

推送本地提交至[远程仓库/分支]，即[origin/master]

```
git push origin master
```

```
(exp_venv) xinyu-yaoer2@ubuntu:~/exp_pyms/exp_pyms$ git pull
Username for 'https://codeup.aliyun.com': 87529617864492384
Password for 'https://87529617864492384@codeup.aliyun.com':
Already up to date.
(exp_venv) xinyu-yaoer2@ubuntu:~/exp_pyms/exp_pyms$ git commit -m "docs: 创建README.md文件"
[master 515a22e] docs: 创建README.md文件
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
(exp_venv) xinyu-yaoer2@ubuntu:~/exp_pyms/exp_pyms$ git push origin master
Username for 'https://codeup.aliyun.com': 87529617864492384
Password for 'https://87529617864492384@codeup.aliyun.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 305 bytes | 305.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://codeup.aliyun.com/6703722449e9309ce56b9b0f/exp_pyms.git
 64a038e..515a22e  master -> master
```

使用 SSH 访问远程仓库

参考文献: [如何配置SSH密钥及自定义SSH认证密钥的路径-云效-阿里云帮助中心](#)

本地环境执行 git pull/push/clone/fetch 等涉及访问远程仓库的命令时，若使用 HTTPS 方式访问，每次均需提供用户名和密码，比较繁琐。一般而言，如果本地开发环境可信，则可以使用**SSH免密认证方式**访问远程仓库，以减少认证交，提高效率。

生成SSH公私钥对

若当前用户 `~/.ssh/` 目录中已存在私钥 `id_rsa` 与公钥 `id_rsa.pub` 文件，直接使用即可。否则使用命令 **ssh-keygen** 新建 SSH 公私钥对，主要参数解析如下：

- `-t` 指定密钥类型，如 `ed25519` 或 `rsa()`
- `-f` 指定用于存储私钥的文件名，若不使用，则提示默认文件及路径为 `~/.ssh/id_rsa`
- `-C` 指定注释内容，可用作公钥名称，Codeup称之为[标题]

这里我使用`ed25519`类型的密钥，使用命令 `ssh-keygen -t ed25519 -f ~/.ssh/id_ed25519 -C "id_ed25519"` 生成公私钥对，如图所示：

```
(exp_venv) xinyu-yaoer2@ubuntu:~/exp_pyms/exp_pyms$ ssh-keygen -t ed25519 -f ~/.ssh/id_ed25519 -C "id_ed25519"
Generating public/private ed25519 key pair.
Created directory '/home/xinyu-yaoer2/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/xinyu-yaoer2/.ssh/id_ed25519
Your public key has been saved in /home/xinyu-yaoer2/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:7vGE8T/wLQttUxkfgVf//RnbF6aYrSnT93HM0io799o id_ed25519
The key's randomart image is:
+--[ED25519 256]--+
|                .o|
|                .o|
|               o..|
|                ++|
|               S  o++|
|              . +..+.o+B|
|               + ==+++.o|
|              . * =0oo++|
|               . +o=0++E|
+-----[SHA256]-----+
```

创建成功后，可在当前用户 `~/.ssh/` 目录查看到新建的公钥文件和密钥文件

```
xinyu-yaoer2@ubuntu:~$ ls ~/.ssh/
id_ed25519  id_ed25519.pub
xinyu-yaoer2@ubuntu:~$
```

配置远程仓库所需公钥

在本地环境使用 `cat` 指令查看公钥

```
xinyu-yaoer2@ubuntu:~$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lbnRpdj0wRnB6aYrSnT93HM0io799o id_ed25519
xinyu-yaoer2@ubuntu:~$
```

在云端Codeup页面右上角选择「个人设置->SSH 公钥」，粘贴上一步复制的公钥后完成添加。

公钥 *

在你添加一个 SSH 公钥 前，你需要[生成它](#)

ssh-rsa AAAAB3NzaG1zb2EAAAADAQABAAQDAK1R

标题 *

your ssh key title

作用范围 *

☒ 全部 (读/写操作包括 clone/pull/push 等) ☐ 只读 (仅支持 clone/pull)

过期时间 ①

请选择日期

添加

重建远程仓库并重新克隆

重复前面的克隆步骤，重新使用SSH方式克隆代码库。

```
xinyu-yaoer2@ubuntu: ~/Beryl_exp_pyms$ git clone git@codeup.aliyun.com:
e$
_pyms.git
Cloning into 'Beryl_exp_pyms'...
The authenticity of host 'codeup.aliyun.com (118.31.165.50)' can't be establishe
d.
RSA key fingerprint is SHA256:yEGmgQNVrc3QAvDvoBrTCF2s07KwmmQ+AbWi9vSt/fE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'codeup.aliyun.com,118.31.165.50' (RSA) to the list o
f known hosts.
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms$ ls
Beryl_exp_pyms
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms$ cd Beryl_exp_pyms/
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ ls -a
. .. .git .gitignore
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$
```

经验证，克隆成功。

代码托管基础练习

进入刚刚创建的本地工作目录执行Git基础练习。

创建文件并同步至远程仓库

在本地工作目录 exp_pyms 中创建 test.py 文件，暂存（Stage）至暂存区后使用 git status 命令查看文件状态。



test.py

```
xinyu-yaoer2@ubuntu: ~/Beryl_exp_pyms/Beryl_exp_pyms
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ echo "import json" > test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git add test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   test.py
```

使用 git commit 命令提交至本地仓库并观察文件状态

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git commit -m "测试1 feat(test.py): 引入json"
[master e27784e] 测试1 feat(test.py): 引入json
1 file changed, 1 insertion(+)
create mode 100644 test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

使用 git push 命令同步至远程仓库并观察文件状态

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 316 bytes | 79.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To codeup.aliyun.com:6703722449e9309ce56b9b0f/Beryl_exp_pyms.git
   dfa50fe..e27784e  master -> master
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

此时，已新建 `test.py` 文件，并完成「暂存、提交、推送」三个基本操作，将本地代码同步托管到云端 Codeup。

修改文件并同步至远程仓库

修改 `test.py` 文件，添加一行代码，然后执行 `git status` 命令查看文件状态。

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ echo "import random" > test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/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:   test.py

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

由此可见，`test.py`在工作录中的状态为被跟踪（Tracked），但修改尚未被暂存（Not Staged），必须再次使用 `git add` 指令暂存该文件，随后方可再次提交。

★另外需要注意的是，该文件已经使用了`add`命令，被`add`的文件就已经被跟踪，再次使用`add`命令，才会将修改的内容再次添加到暂存区。

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git add test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.py
```

随后修订 `test.py`，新增一条语句后再次观察文件状态，可见 `test.py` 同时具有等待被提交以及尚未保存的内容。显然，刚增加的 `import re` 属于尚未被暂存的内容。

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ echo "import re" >> test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.py

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:   test.py
```

这个时候为了简化操作，使用 `git commit` 命令并应用 `-a` 参数提交至本地仓库并观察文件状态。应用 `-a` 时，Git 会自动暂存被修改或删除的文件（未被 Git 跟踪的文件除外）。

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git commit -a -m"feat(test.py): 引入random, re"
[master b4370cf] feat(test.py): 引入random, re
 1 file changed, 2 insertions(+), 1 deletion(-)
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

接着直接使用 `git push` 命令同步至远程仓库并观察文件状态。
在当前本地 `master` 分支使用直接使用 `git push` 等价于 `git push origin master`

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 318 bytes | 318.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To codeup.aliyun.com:6703722449e9309ce56b9b0f/Beryl_exp_pyms.git
   e27784e..b4370cf  master -> master
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

master		创建流水线	克隆/下载
xinyuyaoer	feat(test.py): 引入random, re	b4370cf6	今天 20:36
.gitignore	Initial commit		今天 19:12
test.py	feat(test.py): 引入random, re		今天 20:36

查看远程仓库历史记录确认执行成功。

基于提交历史比较版本差异

`git log` 命令可以用于查看提交历史记录，具有非常强大的能力。此处先练习基本用法。默认输出按时间先后顺序列出所有的提交，最近的更新排在最上面。输出内容中，`commit` 后紧跟的是本次提交的 SHA-1 校验和，可作为提交的版本号

```

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git log
commit b4370cf6c5ca3b4aab8b68f5566bf3e5529590ef (HEAD -> master, origin/master, origin/HEAD)
Author: xinyuyaoer <xinyuyaoer@gmail.com>
Date: Tue Oct 8 05:36:32 2024 -0700

    feat(test.py): 引入random, re

commit e27784e25354423532fe4d65c4845b95c417294b
Author: xinyuyaoer <xinyuyaoer@gmail.com>
Date: Tue Oct 8 05:10:08 2024 -0700

    测试1 feat(test.py): 引入json

commit dfa50fe076e2b7b5594fa8aea7cd328cd37babb3
Author: 歆毓xinyu <3049736719@qq.com>
Date: Tue Oct 8 19:12:19 2024 +0800

    Initial commit

```

使用 -p 或 --patch 参数查看每次提交所引入的差异，并在其后指定要查看的提交数，如 -1 说明只查看最近的 1 次提交。

```

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git log -p -1
commit b4370cf6c5ca3b4aab8b68f5566bf3e5529590ef (HEAD -> master, origin/master, origin/HEAD)
Author: xinyuyaoer <xinyuyaoer@gmail.com>
Date: Tue Oct 8 05:36:32 2024 -0700

    feat(test.py): 引入random, re

diff --git a/test.py b/test.py
index 0349a44..3708c1c 100644
--- a/test.py
+++ b/test.py
@@ -1,2 @@
-import json
+import random
+import re

```

实际编程时，经常会遇到希望将当前版本与过往提交的某个版本进行比较的场景。此时可在 git log 查询的基础上使用 **git diff** 查询某过往提交版本与当前工作目录中版本的差异。

- 比较当前版本与指定版本的差异

```
git diff <commit>
```

- 比较指定版本与指定版本的差异

```
git diff <commit1> <commit2>
```


- 比较当前版本与指定版本的差异，仅显示差异部分

```
git diff <commit> --stat
```

实验中，先修改test.py，然后将test.py当前工作目录中的版本与SHA-1校验和为「394ea07...」的历史提交版本比较。

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ echo "print("Hello,world!");" >> test.py
bash: !": event not found
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ echo "print('Hello, world!')" >> test.py
bash: !': event not found
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ echo 'print("Hello, world!")' >> test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git diff e27784e25354423532fe4d65c4845b95c417294b test.py
diff --git a/test.py b/test.py
index 0349a44..21e0670 100644
--- a/test.py
+++ b/test.py
@@ -1,3 @@
-import json
+import random
+import re
+print("Hello, world!")
```

从图中可以发现，在 Bash 中，! 有特殊含义，如果它不是在特定的命令历史扩展等上下文中正确使用，就会出现“event not found”错误。

解决方案：①使用!转义 ②里面的内容使用单引号括起来，即：echo 'print("Hello, world!")' >> test.py

解释😁：这是因为在 Bash 中，使用双引号时，Bash 会尝试对一些特殊字符（如 \$、!、* 等）进行解析和扩展。而使用单引号时，Bash 会将单引号中的内容视为完全的字面字符串，不进行任何特殊字符的解析和扩展。

echo "print("Hello,world!");" >> test.py 这个命令中，双引号内的字符串包含了双引号和括号，Bash 可能会尝试对其进行不适当的解析，导致错误。而 echo 'print("Hello, world!")' >> test.py 中，单引号确保了整个字符串被视为纯文本，不会触发 Bash 的特殊字符解析机制，所以能够正确执行。

按需撤销暂存、提交或推送

- 撤销暂存

查看文件状态，确认 test.py 修订为暂存，随后执行暂存并再次查看文件状态


```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/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:   test.py

no changes added to commit (use "git add" and/or "git commit -a")
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git add test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.py
```

此时可见系统提示 (use "git restore --staged ..." to unstage), 应用该指令后观察文件状态, 发现 `test.py` 状态恢复为尚未暂存(Modified by Not Staged)。该命令的本意是[To restore a file in the index to match the version in HEAD]

所以接下来输出下面的指令:

```
git restore --staged <file> # 将提交区(代码仓库)中HEAD指向的版本复制到暂存区
git status # 查看状态
```

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git restore --staged test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/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:   test.py

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

也可以使用命令 `git reset HEAD` 来取消对 `test.py` 的暂存。需要特别注意的是, 若该指令重复执行, 将会**依次取消**对于`test.py`的历史提交, 是非常危险的命令, **必须慎用**。

```
git add test.py # 重新暂存文件
git status # 查看状态
git reset HEAD test.py # 取消暂存
git status # 查看状态
```

```

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git add test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.py

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git reset HEAD test.py
Unstaged changes after reset:
M       test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/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:   test.py

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

```

• 撤销提交

提交当前包含打印 Hello, world! 语句的 `test.py` 并查看文件状态。

```

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git add test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.py

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git commit -a -m "feat(test.py): 打印Hello, world!"
[master c64a511] feat(test.py): 打印Hello, world!
 1 file changed, 1 insertion(+)
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

```

使用命令 `git reset HEAD`~ 撤销提交后查看文件状态, 可发现 `test.py` 已恢复到尚未暂存与提交的状态。

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git reset HEAD~
Unstaged changes after reset:
M      test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/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:   test.py

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

★注意：如若 git reset HEAD~重复执行，会造成 HEAD 不断偏移，谨慎使用！

• 撤销推送

本地回滚 HEAD 后强制推送远程仓库，等价于撤销先前的推送。

但是经过了解，本地回滚 HEAD 后强制推送远程仓库在一定程度上可以使远程仓库状态恢复到较早版本，类似撤销先前推送的部分效果。但这种操作可能会给协同开发者带来严重冲突，且并非真正意义上的撤销推送，只是用旧版本覆盖新版本，应谨慎使用并提前与团队成员沟通协调。

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git commit -a -m "feat(test.py): 打印Hello, world!"
[master 48e8168] feat(test.py): 打印Hello, world!
 1 file changed, 1 insertion(+)
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git push origin master
ssh: connect to host codeup.aliyun.com port 22: Connection refused
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 352 bytes | 117.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To codeup.aliyun.com: [redacted]_l_exp_pyms.git
 b4370cf..48e8168  master -> master
```

执行以上操作后登陆Codeup，查看远程仓库 master 分支「提交」页面，可见本次新提交记录。

xi

xinyuyaoer feat(test.py): 打印Hello, world!

48e81689

今天 00:04

文件 | Blame

47 Bytes · 3 lines

```

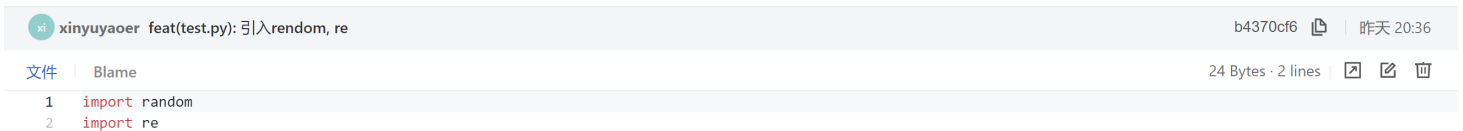
1  import random
2  import re
3  print("Hello, world!")
```

输入以下指令用来撤销推送：

```
git reset HEAD~ # 撤销提交 (本地回滚 HEAD)
git push origin master --force # 强制推送到远程仓库
#或者用: git push -f origin master # 强制推送到远程仓库
```

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git reset HEAD~
Unstaged changes after reset:
M   test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git push -f origin master
Total 0 (delta 0), reused 0 (delta 0)
To codeup.aliyun.com:([REDACTED])_exp_pyms.git
+ 48e8168...b4370cf master -> master (forced update)
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$
```

执行以上操作后登陆 Codeup，查看远程仓库 master 分支「提交」页面，可见本次新提交记录。



注意：因为 git push -f 会强行推送，并抹去上次推送的全部内容非常危险😱！强制推送应该谨慎使用，因为它可能会导致其他从该远程仓库拉取代码的人出现问题，覆盖他们的工作成果并造成混乱。如果可能，应该尽量通过更安全的方式（如先拉取远程分支进行合并等）来同步代码，而不是频繁使用强制推送。

删除文件以及恢复指定文件

保留文件，取消跟踪

```
git status # 查看现在的状态，发现有部分内容没有被add，即没有被跟踪
git add test.py # 全部跟踪
git status
git rm --cached test.py # 取消对test.py的跟踪
git status
git add test.py # 再次跟踪
git status
```

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/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:   test.py

no changes added to commit (use "git add" and/or "git commit -a")
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git add test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.py

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git rm --cached test.py
rm 'test.py'
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    test.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test.py

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git add test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.py
```

从暂存区域恢复文件

修订文件，暂存，随后删除文件**但并不暂存删除操作**，可将暂存区域 index 暂存的版本恢复到工作目录。

```
rm test.py # 删除文件,但并不暂存删除操作
```

```
git restore test.py # 从暂存区恢复刚刚删除的test.py文件，即将暂存区的版本复制到本地工作目录
```



```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ echo 'print("add something for git_exp");' >> testst.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ echo 'print("add something for git_exp");' >> test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git giff test.py
git: 'giff' is not a git command. See 'git --help'.
```

The most similar command is
diff

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git diff test.py
diff --git a/test.py b/test.py
index 21e0670..3c75ab9 100644
```

```
--- a/test.py
+++ b/test.py
@@ -1,3 +1,4 @@
```

```
import random
import re
print("Hello, world!")
+print("add something for git_exp");
```

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.
```

Changes to be committed:
(use "git restore --staged <file>..." to unstage)
modified: test.py

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: test.py

Untracked files:
(use "git add <file>..." to include in what will be committed)
testst.py

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git add test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ rm testst.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.
```

Changes to be committed:
(use "git restore --staged <file>..." to unstage)
modified: test.py

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ rm test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ cat test.py
cat: test.py: No such file or directory
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.
```

Changes to be committed:
(use "git restore --staged <file>..." to unstage)
modified: test.py

Changes not staged for commit:
(use "git add/rm <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)
deleted: test.py

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git restore test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ cat test.py
import random
import re
print("Hello, world!")
print("add something for git_exp");
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.
```

Changes to be committed:
(use "git restore --staged <file>..." to unstage)
modified: test.py

从本地仓库恢复文件

修订文件，暂存，随后删除文件且同时暂存删除操作，则先前暂存但未提交的修订内容丢失，仅可从本地仓库 HEAD 恢复。

```
git status # 查看状态
git rm test.py # 使用命令rm删除文件后，也删除状态
# 使用上面的命令，会发现有报错如下：
# error: the following file has changes staged in the index: ...
# 对于已经暂存修改的文件，使用git rm删除时必须带上参数-f
git rm test.py -f
git status # 查看状态，发现test.py已被标记为deleted，先前暂存的modified消失了，此时暂存区已经没有test
git restore test.py # 执行restore不能恢复被删除的文件，思考为什么？ ---因为暂存区里面的想要被恢复的内容
git restore --staged test.py # 将版本库（提交（commit）区）中HEAD指向的版本复制到暂存（add）区
cat test.py # 查看test.py并没有恢复
git status # 查看状态
git restore test.py # 从暂存区恢复test.py文件，即将暂存区的版本复制到本地工作目录中
cat test.py # 查看test.py的内容，看看是哪一个版本，发现是不包含 Hello world! 和add something for git
```

```

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.py

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git rm test.py
error: the following file has changes staged in the index:
        test.py
(use --cached to keep the file, or -f to force removal)
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git rm test.py -f
rm 'test.py'
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    test.py

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git restore test.py
error: pathspec 'test.py' did not match any file(s) known to git
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git restore --staged test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ cat test.py
cat: test.py: No such file or directory
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    test.py

no changes added to commit (use "git add" and/or "git commit -a")
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git restore test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ cat test.py
import random
import re
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

```

当然我们也可以使用单条命令如下，基于提交区 HEAD 指向的版本，同时恢复暂存区与工作区中的 `test.py`。

`git restore --source=HEAD --staged --worktree test.py` # 该命令等价于 `git checkout HEAD test.py`

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git rm test.py
rm 'test.py'
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    deleted:    test.py

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git restore --source=HEAD --staged --worktree test.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ cat test.py
import random
import re
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git rm test.py
rm 'test.py'
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    deleted:    test.py

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git checkout HEAD test.py
Updated 1 path from 10ac8b2
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ cat test.py
import random
import re
```

从远程仓库恢复文件

若删除的文件因各种原因已无法从本地仓库及缓冲区恢复，则需考虑从远程仓库恢复文件。

命令 `git fetch origin` 会抓取克隆(或上一次抓取)后新推送的所有工作。 特别注意 `git fetch` 只会将数据下载到本地仓库，而不会自动合并或修改当前工作去的文件，后续需要手动融合。

运行 `git pull` 则会首先自动抓取远程分支的所有工作，并将其融合至当前分支。

具体而言：

- 从远程仓库下载最新的提交历史和代码文件到本地；
- 若本地修改尚未提交，会尝试将远程代码和本地代码自动合并
- 若合并过程中出现冲突，需要用户手动解决冲突并提交合并结果。

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git pull
Already up to date.
```

在正常情况下，提示均已更新至最新版本。

这个指令在多人协作场景发挥重要作用，要牢记!!! 🤖

分支管理训练

在Git中，分支(Branch)是指在版本控制下独立开发的代码线。每个Git仓库都可以有一个或多个分支，每个分支可以包含不同的代码修改和提交历史。

分支在Git中非常灵活和强大，它们提供了以下好处：

- a. **并行开发**:可以在不同的分支上同时进行独立开发，提高团队工作效率
- b. **特性开发**:可以在单独的分支上开发新的功能，然后在完成后将其合并到主分支。
- c. **Bug修复**:可以在一个分支上修复错误，然后将修复内容合并到主分支，而不影响正在进行的其他开发工作。
- d. **版本管理**:每个分支都有自己的提交历史，可以方便地在不同的版本之间切换和查看。

创建 develop 分支用于开发

【💡 理论回顾】创建一个分支，本质上就是创建了一个指向当前提交对象的指针。

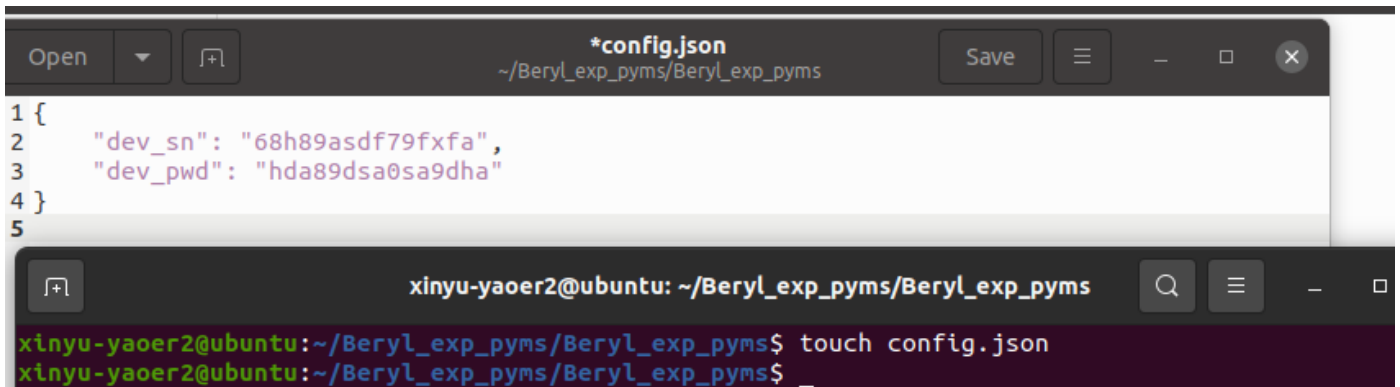
1. 使用以下指令新建并切换到 develop 分支里面。

```
git branch develop # 该命令本质上是在当前所在的提交对象上创建一个新的指针
git switch develop # 切换到develop分支上
```

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git branch develop
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git switch develop
Switched to branch 'develop'
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch develop
nothing to commit, working tree clean
```

此外也可以使用更加简洁的命令 `git checkout -b <branch>` 可以达到以上同样的效果

2. 创建文件config.json,仅供本地测试使用，需要通过.gitignore文件忽略该文件。



The screenshot shows a code editor window with a file named `*config.json` open. The file content is as follows:

```
1 {
2   "dev_sn": "68h89asdf79fxfa",
3   "dev_pwd": "hda89dsa0sa9dha"
4 }
5
```

Below the code editor, a terminal window is shown with the following commands and output:

```
xinyu-yaoer2@ubuntu: ~/Beryl_exp_pyms/Beryl_exp_pyms
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ touch config.json
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$
```

3. 创建文件test_br.py, 其中包含需要Git托管的业务逻辑代码。



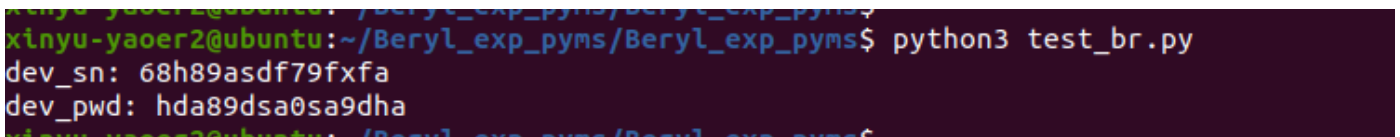
```
test_br.py
~/Beryl_exp_pyms/Beryl_exp_pyms

1 import json
2 def get_config(path: str) -> dict:
3     with open(path, "r") as f:
4         return json.load(f)
5 if __name__ == '__main__':
6     config: dict = get_config("./config.json")
7     print(f"dev_sn: {config['dev_sn']}")
8     print(f"dev_pwd: {config['dev_pwd']}")
9

xinyu-yaoer2@ubuntu: ~/Beryl_exp_pyms/Beryl_exp_pyms

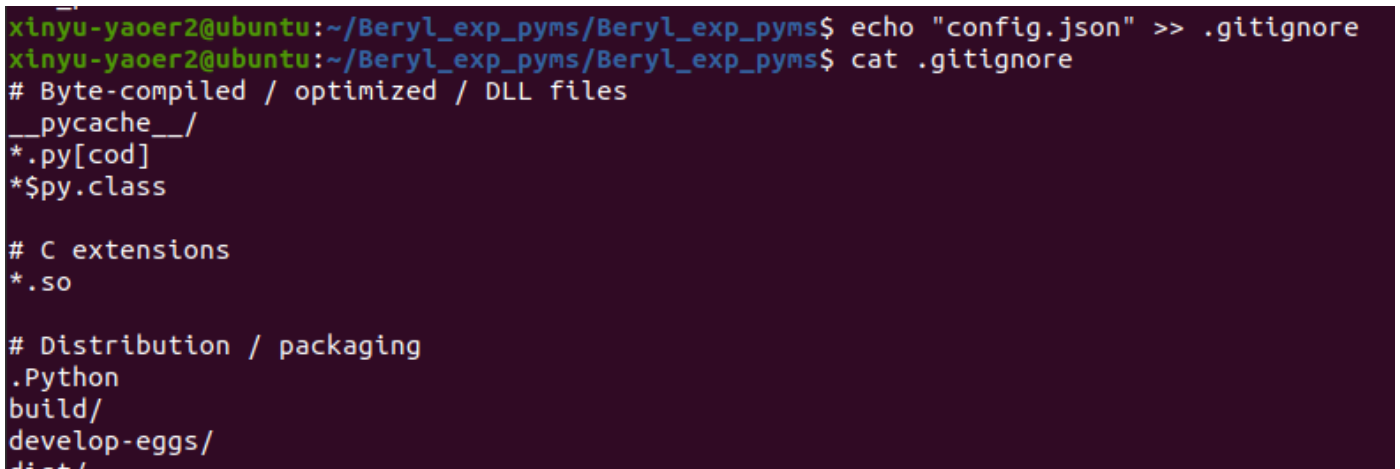
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ touch test_br.py
```

4. 本地测试Python脚本



```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ python3 test_br.py
dev_sn: 68h89asdf79fxfa
dev_pwd: hda89dsa0sa9dha
```

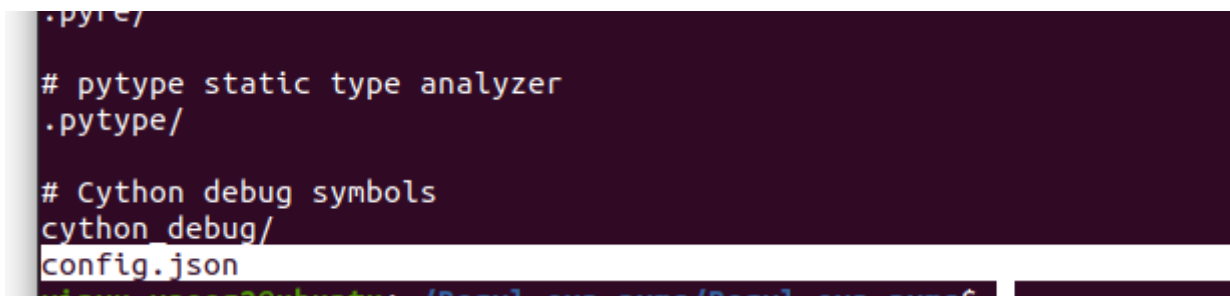
5. 配置忽略 config.json , 即将 config.json 写入 .gitignore 文件。



```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ echo "config.json" >> .gitignore
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ cat .gitignore
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
```



```
.pyc/

# pytype static type analyzer
.pytype/


# Cython debug symbols
cython debug/
config.json
```

6. 暂存、提交、推送。完成以下操作后，在Codeup查看 develop 分支，可发现 test_br.py 已经成功推送到远程仓库。

```
git add test_br.py .gitignore # 跟踪新文件, 暂存修订文件
git status # 观察文件状态
git commit -m "feat(test_br.py): 读取config.json, 打印dev_sn与dev_pws; docs (.gitignore) : 添加config.json"
git status # 观察文件状态
git push origin develop # 推送到远程仓库/分支[origin/develop], 若远程分支不存在, 则创建
```

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git add test_br.py .gitignore
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   .gitignore
        new file:   test_br.py

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git commit -m "feat(test_br.py): 读取config.json,
打印dev_sn与dev_pws; docs (.gitignore) : 添加config.json"
[develop 83a1fc1] feat(test_br.py): 读取config.json, 打印dev_sn与dev_pws; docs (.gitignore) : 添加config.json
2 files changed, 10 insertions(+)
create mode 100644 test_br.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch develop
nothing to commit, working tree clean
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git push origin develop
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 601 bytes | 200.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0)
To codeup.aliyun.com:6703722449e9309ce56b9b0f/Beryl_exp_pyms.git
 * [new branch]      develop -> develop
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$
```

 develop 在今天 19:49被更新

 develop ▾

 xinyuyaoer feat(test_br.py): 读取config.json, 打印dev_sn与dev_pws; docs (.gitignore) : 添加config.json

 .gitignore

 test.py

 test_br.py

创建 feature 分支用于特定功能开发

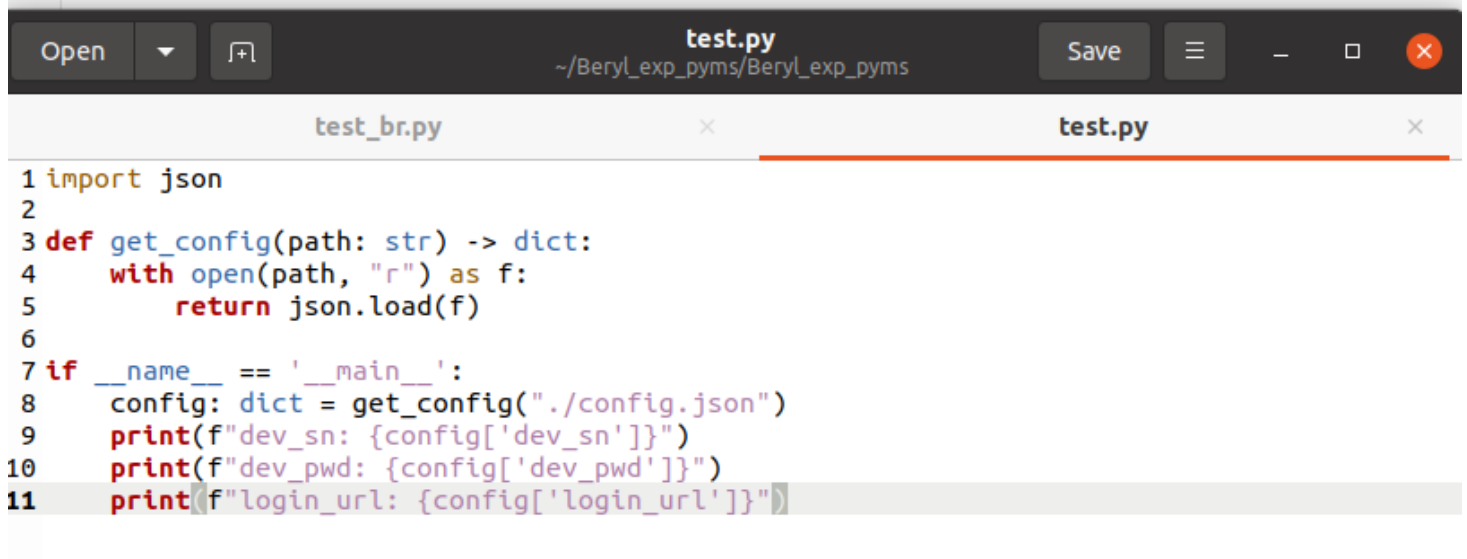
1. 从当前分支创建 feature 分支并推送当前暂存文件至远程仓库对应分支。

```
git checkout -b feature # 等价于 git branch feature \n git switch feature
```



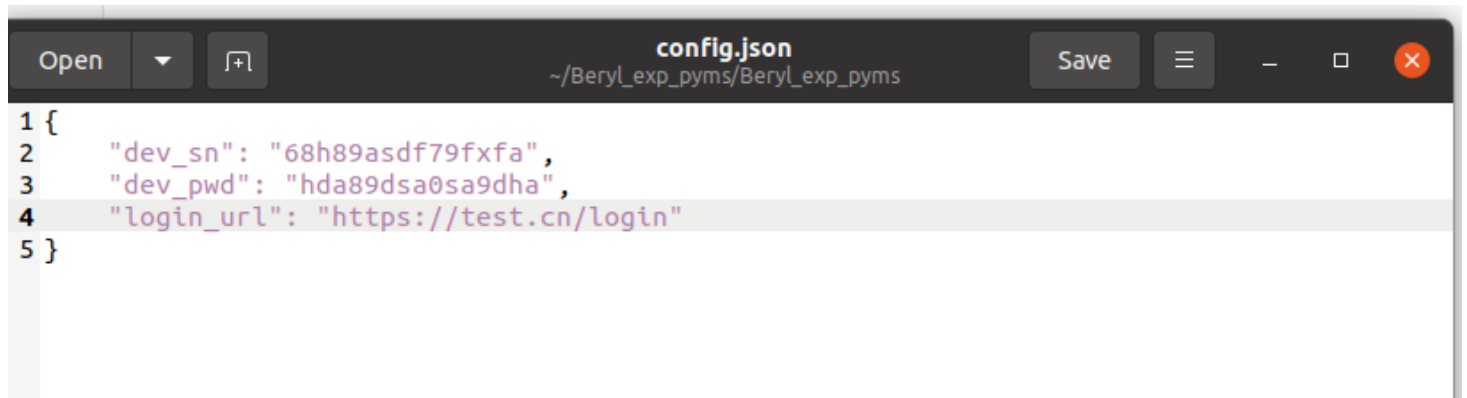
```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git checkout -b feature
Switched to a new branch 'feature'
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$
```

2. 修订 test.py 中的代码，完成 feature 功能需求，从 config.json 读取并打印 login_url 。

A screenshot of a code editor window titled 'test.py' with the path '~/Beryl_exp_pyms/Beryl_exp_pyms'. The editor shows the following Python code:

```
1 import json
2
3 def get_config(path: str) -> dict:
4     with open(path, "r") as f:
5         return json.load(f)
6
7 if __name__ == '__main__':
8     config: dict = get_config("./config.json")
9     print(f"dev_sn: {config['dev_sn']}")
10    print(f"dev_pwd: {config['dev_pwd']}")
11    print(f"login_url: {config['login_url']}")
```

3. 修订 config.json 对应增补 login_url 相关内容，用于 test.py 测试。

A screenshot of a code editor window titled 'config.json' with the path '~/Beryl_exp_pyms/Beryl_exp_pyms'. The editor shows the following JSON content:

```
1 {
2     "dev_sn": "68h89asdf79fxfa",
3     "dev_pwd": "hda89dsa0sa9dha",
4     "login_url": "https://test.cn/login"
5 }
```

4. 执行本地测试，确认 feature 开发情况满足需求

```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ python3 test.py
dev_sn: 68h89asdf79fxfa
dev_pwd: hda89dsa0sa9dha
login_url: https://test.cn/login
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$
```

5. 本地暂存提交，推送远程仓库 feature 分支

```

rinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch feature
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:   test.py

no changes added to commit (use "git add" and/or "git commit -a")
rinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git add test.py
rinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch feature
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.py

rinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git commit -m "feat(test.py):打印login_url"
[feature b8e8dcb] feat(test.py):打印login_url
1 file changed, 11 insertions(+), 2 deletions(-)
rinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch feature
nothing to commit, working tree clean
rinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git push origin feature
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 499 bytes | 499.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To codeup.aliyun.com:6[redacted]p_pyms.git
 * [new branch]   feature -> feature
rinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch feature
nothing to commit, working tree clean
rinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$

```

6. 将 feature 分支合并 (Merge) 到 develop 分支，并推送到远程仓库

```

git checkout develop # 切换到develop分支
cat test.py # 此时develop分支下的test.py还是前面最原始的内容
git merge feature # 把feature里面的内容合并到develop里面，如果遇到同名的进行覆盖
git status
cat test.py # 此时develop分支下的test.py 已经存在打印login_ur1的语句
git push origin develop

```

```

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git checkout develop
Switched to branch 'develop'
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ cat test.py
import random
import re
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git merge feature
Updating 83a1fc1..b8e8dcb
Fast-forward
 test.py | 13 ++++++++--
 1 file changed, 11 insertions(+), 2 deletions(-)
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git status
On branch develop
nothing to commit, working tree clean
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ cat test.py
import json

def get_config(path: str) -> dict:
    with open(path, "r") as f:
        return json.load(f)

if __name__ == '__main__':
    config: dict = get_config("./config.json")
    print(f"dev_sn: {config['dev_sn']}")
    print(f"dev_pwd: {config['dev_pwd']}")
    print(f"login_url: {config['login_url']}")
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git push origin develop
Total 0 (delta 0), reused 0 (delta 0)
To codeup.aliyun.com:67c[REDACTED]/Beryl_exp_pyms.git
 83a1fc1..b8e8dcb develop -> develop

```

此时查看 Codeup，在 Beryl_exp_pyms 项目的 develop 分支，可发现代码已提交成功若确认开发完成，则可以继续将 develop 分支合并到 master 分支并推送远程仓库。若云端 master 有对应生产环境流水线，此时便会触发流水线执行打包部署。

```

git checkout master # 切换到master分支
git merge develop
git push origin master

```

```

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git merge develop
Updating b4370cf..b8e8dcb
Fast-forward
 .gitignore | 1 +
 test.py    | 13 ++++++++--
 test_br.py | 9 ++++++++
 3 files changed, 21 insertions(+), 2 deletions(-)
 create mode 100644 test_br.py
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git push origin master
Total 0 (delta 0), reused 0 (delta 0)
To codeup.aliyun.com:[REDACTED]Beryl_exp_pyms.git
 b4370cf..b8e8dcb master -> master
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$

```

创建 hotfix 分支执行热修复

代码在 master 上线发布后，若突然发现存在严重 bug，比如「不应打印密码!」，则需马上修复，可立即从 master 分支签出一个 hotfix 分支用于修复处理。

```
git checkout -b hotfix
```

本实验中，简单设定修复措施，即删除 test.py 打印 dev_pwd 的语句

```
1 import json
2
3 def get_config(path: str) -> dict:
4     with open(path, "r") as f:
5         return json.load(f)
6
7 if __name__ == '__main__':
8     config: dict = get_config("./config.json")
9     print(f"dev_sn: {config['dev_sn']}")
10    print(f"dev_pwd: {config['dev_pwd']}")
11    print(f"login_url: {config['login_url']}")
```

完成修订测试后，将 hotfix 分支合并到 master 分支和 develop 分支

```
1 $ git commit -a -m "fix(test.py): 不应打印密码, 删除对应语句"
2 ...
3 $ git checkout master      # 切换到分支master
4 ...
5 $ git merge hotfix         # 在分支master合并hotfix
6 Updating d12d55b..bb4c71b
7 Fast-forward
8 test.py | 1 -
9 1 file changed, 1 deletion(-)
10 $ cat test.py              # 自行观察, 此时打印 dev_pwd 的语句已删除
11 ...
12 $ git push origin master   # 推送远程仓库分支master
13 ...
14 $ git checkout develop     # 切换到分支develop
15 ...
16 $ git merge hotfix         # 在分支develop合并hotfix
17 ...
18 $ git push origin develop  # 推送远程仓库分支develop
19 ...
```

操作如下:

```

xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git checkout -b hotfix
Switched to a new branch 'hotfix'
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git commit -a -m "fix(test.py): 不应打印密码，删除对应语句"
[hotfix a9a6a40] fix(test.py): 不应打印密码，删除对应语句
 1 file changed, 3 deletions(-)
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git merge hotfix
Updating b8e8dc..a9a6a40
Fast-forward
 test.py | 3 ---
 1 file changed, 3 deletions(-)
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ cat test.py
import json
def get_config(path: str) -> dict:
    with open(path, "r") as f:
        return json.load(f)
if __name__ == '__main__':
    config: dict = get_config("./config.json")
    print(f"dev_sn: {config['dev_sn']}")
    print(f"login_url: {config['login_url']}")
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 407 bytes | 407.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To codeup.aliyun.com:6[REDACTED]xp_pyms.git
 b8e8dc..a9a6a40 master -> master
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git checkout develop
Switched to branch 'develop'
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git merge hotfix
Updating b8e8dc..a9a6a40
Fast-forward
 test.py | 3 ---
 1 file changed, 3 deletions(-)
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git push origin develop
Total 0 (delta 0), reused 0 (delta 0)
To codeup.aliyun.com:6[REDACTED]of/Beryl_exp_pyms.git
 b8e8dc..a9a6a40 develop -> develop

```

最后删除 hotfix 分支

```
git branch -d hotfix
```



```
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git branch -r
origin/HEAD -> origin/master
origin/develop
origin/feature
origin/master
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git branch
* develop
feature
hotfix
master
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git branch -d hotfix
Deleted branch hotfix (was a9a6a40).
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git branch -r
origin/HEAD -> origin/master
origin/develop
origin/feature
origin/master
xinyu-yaoer2@ubuntu:~/Beryl_exp_pyms/Beryl_exp_pyms$ git branch
* develop
feature
master
```

