README

用 GitHub Desktop 共享代码

在桌面上下载一个GitHub Desktop注册登录一个Github账号

邀请组员进仓库：

打开网页版 GitHubhttps://github.com/注册登录账号，注册完登录进去后给我你们的邮箱账号和用户名我拉你们进仓库。

在 GitHub Desktop 的操作：

登录自己的账号：在 GitHub Desktop 登录自己的 GitHub 账号。

克隆仓库：点击 `File` -> `Clone Repository`。

自动出现：在 `GitHub.com` 列表里，会直接看到的 `csle-collab` 仓库。

克隆：点一下 `Clone` 按钮，代码就同步到他们的电脑上了

上传（Push）：

- 在 GitHub Desktop 左侧会看到新文件，在底部的 Summary 里写“Add project bridge and config”。
- 点击 **Commit to main**，然后点击顶部的 **Push origin**。

虚拟机镜像（OVA）的传输与配置

1. **导入**：打开 VMware，选择 `文件 -> 打开`，选择收到的 `.ova` 或 `.ovf`。
2. **重命名**：建议导入时将虚拟机重命名，以区分原有环境。
3. **网络冲突处理**：
   - 导入过程中，如果弹出 **“MAC 地址策略”** 选项，请选择 **“重新生成所有网卡的 MAC 地址”**。
   - 进入系统后，如果无法联网，请执行：`sudo dhclient`。
4. **权限修复**：进入系统后，先执行 `sudo chmod -R 777 /home/li/csle` 确保所有脚本有读写权限。

> **核心文件路径备忘**
>
> - **补丁文件**：`~/csle/simulation-system/venv/lib/python3.10/site-packages/csle_common/metastore/metastore_facade.py`
> - **虚拟环境激活**：`source ~/csle/simulation-system/venv/bin/activate`
> - **手动数据生成脚本**：`python3 ~/csle/create_fake_data.py`

账号与凭据

用户登录密码：

- **用户名**：`li`
- **密码**：`ljy328`（这个最重要，进系统、执行 `sudo` 命令全靠它）。

数据库密码：

- **数据库用户名**：`postgres`

- **密码**：通常也是 `postgres`
- 组员如果直接进数据库改数据，必须知道这个。

网页登录密码 (CSLE UI)：

- **登录账号**：`admin` 。
- **密码**：`admin`。

虚拟机的操作

操作流程：

终端 1：启动后台容器

```
cd ~/csle
sudo docker-compose up -d
# 必须停止自带的 API，由我们修改后的 mini_app 接管
sudo docker stop csle-rest-api
```

终端 2：启动补丁版 API 桥接器

```
cd ~/csle
source ./simulation-system/venv/bin/activate
# 现在的 mini_app.py 已支持绕过解析报错
python3 mini_app.py
```

终端 3：启动管理后台

```
cd ~/csle/management-system/csle-mgmt-webapp/build
python3 -m http.server 3000
```

csle网址：http://localhost:8080

后端JSON网址：http://localhost:8080/simulations

（**重点在后端能够刷新出来东西，说明数据库这部分我做通了**）

如何运行仿真 (Role B 指南)

当环境稳定后，(B) 需执行以下操作：

脚本位置：进入 `~/csle/examples/data_collection/static_sequences/level_9/`。

执行命令：

```
# 不再强制设置 METASTORE_DISABLED=True，因为我们已经修复了 Metastore 转换逻辑
python3 run.py
```

故障排除：若运行报错 `KeyError` 或 `TypeError`，说明 CSLE 源码又有新的嵌套字段未被 Mock。请立即通知 Role A (Li)，我们在 `metastore_facade.py` 中补充默认值。

数据产出与保存

仿真运行后，数据会产生在三个地方：

- **轨迹数据 (Trajectories)：** 自动保存在 PostgreSQL 数据库的 `simulations` 表中。
- **实时指标 (Kafka)：** 仿真过程中，实时性能指标会推送到 Kafka 的 `csle-emulation-statistics` 主题。
  - *查看命令：* `sudo docker exec -it csle-kafka /bin/bash -c "kafka-console-consumer --bootstrap-server localhost:9092 --topic csle-emulation-statistics"`
- **IDS 日志：** 若开启了仿真容器，Snort/Suricata 日志会出现在 `/var/log/csle/` 下。

轨迹导出 (Role B 专用)

轨迹查看：仿真运行后，数据将写入数据库。

CSV 导出脚本：

```
# 专门为 Role B 准备的导出补丁
from csle_common.metastore.metastore_facade import MetastoreFacade
# 经过 Li 修复后的接口，现在可以直接拿数据
executions =
MetastoreFacade.list_emulation_executions_for_a_given_emulation("csle-level9-
070")
print(f"成功获取 {len(executions)} 条仿真轨迹！")
```

安全停止集群

实验结束后，请务必按以下顺序关闭，防止数据库文件损坏：

停止进程：各个终端按 `Ctrl + C`。

清理 Docker：`cd ~/csle && sudo docker-compose stop`。

释放端口：`sudo fuser -k 8080/tcp` (针对 API) 和 `sudo fuser -k 3000/tcp` (针对网页)。

关于“转圈圈”问题的特别说明

现象：点击菜单一直转圈。

根源：前端请求的 JSON 结构与原始数据库不匹配。

对策：

1. 现在的 `mini_app.py` 已经硬编码了“保底数据”。
2. **强制刷新**：在浏览器按下 `Ctrl + F5` 清除缓存。
3. **查看进度**：如果网页不动，直接看 **终端 2** 的日志输出，只要看到 `200 OK`，说明数据已成功送达。

（**这部分问题还未彻底解决，属于前端问题暂时不在我的范围内，前端的修复先往后放**）

给组员的温馨提示

> “我们现在使用的系统经过了底层源码级的‘热修补’。如果你的运行报错与我的不同，请不要尝试重新安装 `csle` 库，因为那会覆盖掉我已经修好的 `metastore_facade.py` 补丁。遇到问题，先同步最新的 `venv` 或联系我手动修复。”

对了，另外一提，最近正在调的指令和配置文件命令如下（Role B训练所需的标准轨迹csv）：

```
(venv) li@li-virtual-machine:~/csle$ nano /home/li/csle/simulation-
system/venv/lib/python3.10/site-
packages/csle_common/metastore/metastore_facade.py
^Z(venv) li@li-virtual-machine:~/csle$ python3
/home/li/csle/examples/data_collection/static_sequences/level_9/run.py
```

这两条指令我正在修，有小段时间了，需要加的东西有点多，我先把虚拟机给你们这两条指令我会在过程中接着修，最后改完的完整文件我到时候给你们放Github Desktop里面，有问题随时找我

应急数据产出方案 (备选方案)

> 如果原生仿真脚本由于库版本问题依然报 `KeyError`，请执行以下脚本直接生成 Role B 训练所需的标准轨迹 CSV：

```
# 运行 Li 编写的数据模拟脚本
python3 ~/csle/create_fake_data.py
```

生成的 CSV 位于：`~/csle/traces_output_csv/`，已对标 CSLE 标准格式。（这个我已经生成好了可以用来应急）

> 注意：那个计划也只能做大致参考，我们现在主要是把环境搭出来能跑案例就行，计划里面的一些让我们跑py文件可能我们的电脑里并没有该案例，可以和AI沟通一下通过find指令找到我们文件夹里面真是存在的案例作为平替来运行作为判断环境是否搭建完成的一个检验。