预期目标

- 将项目升级至 Python 3.9+、Fabric 2.5.0、PostgreSQL 17，Elasticsearch7.x+（测试用的为8.x），操作系统为 Linux CentOS 8。

目录结构

- `proj/`：Celery 应用与任务
- `fabfile.py`：Fabric 2 任务（启动/停止 workers、批量入队、查询、清理）
- `requirements.txt`：Python 依赖1. 系统准备（CentOS 8）

0. 换源到 Vault（CentOS 8 已 EOL，必须先切换）

```
sudo sed -i '/mirrorlist=/s/^/#/' /etc/yum.repos.d/CentOS-*.repo
sudo sed -i '/#baseurl=/s/^#//' /etc/yum.repos.d/CentOS-*.repo
sudo sed -i 's|mirror.centos.org|vault.centos.org|g' /etc/yum.repos.d/CentOS-
*.repo
sudo dnf clean all
sudo dnf makecache
```

1. 基础工具

```
sudo dnf -y update
sudo dnf -y install git gcc gcc-c++ make openssl-devel libffi-devel bzip2-devel
libpq-devel
# 若某些开发包缺失，可启用 PowerTools：
# sudo dnf config-manager --set-enabled powertools
```

2. 我碰到的问题，供参考（非必须）

- Sudo 权限不足：使用 root 将用户加入 `wheel` 组并启用 sudoers 中的 `%wheel`。

  ```
  su -
  usermod -aG wheel <你的用户名>
  visudo   # 确认：%wheel  ALL=(ALL)  ALL
  ```

- dnf 被锁/僵尸进程：停止 PackageKit，清理锁并重建缓存。

  ```
  sudo systemctl stop packagekit && sudo systemctl disable packagekit && sudo
  systemctl mask packagekit
  sudo pkill -9 packagekitd pk-command-not-found || true
  sudo pkill dnf rpm || true; sleep 2; sudo pkill -9 dnf rpm || true
  sudo rm -f /run/dnf.pid /var/run/dnf.pid /var/lib/rpm/.rpm.lock 2>/dev/null
  sudo dnf clean all && sudo dnf makecache
  ```

- 网络/DNS：确认网卡自启与 DNS。

```
ip a
ping -c 4 8.8.8.8 || true
ping -c 4 www.baidu.com || true
# DNS 兜底：
echo -e "nameserver 8.8.8.8\nnameserver 223.5.5.5" | sudo tee
/etc/resolv.conf
# 网卡自启：
nmcli connection show
sudo nmcli connection modify <你的连接名> connection.autoconnect yes
sudo nmcli connection up <你的连接名>
sudo systemctl restart NetworkManager
```

3. 安装 Python 3.9 + venv（CentOS 8）

```
sudo dnf -y module enable python39
sudo dnf -y module install python39
python3.9 -V
python3.9 -m venv venv
source venv/bin/activate
pip install --upgrade pip
```

说明：在 `venv` 中执行 `sudo dnf`/`systemctl` 不受影响；venv 仅影响 Python/pip 路径。

4. 安装 RabbitMQ 与 Redis

```
# Erlang（RabbitMQ 依赖）
curl -s https://packagecloud.io/install/repositories/rabbitmq/erlang/script.rpm.sh
| sudo bash
sudo dnf -y install erlang

# RabbitMQ
curl -s https://packagecloud.io/install/repositories/rabbitmq/rabbitmq-
server/script.rpm.sh | sudo bash
sudo dnf -y install rabbitmq-server
sudo systemctl enable --now rabbitmq-server

# Redis
sudo dnf -y install redis
sudo systemctl enable --now redis
```

注意：上述 `script.rpm.sh` 是仓库安装脚本，需用 bash 执行，非 RPM 包。

默认连接串：RabbitMQ（broker）`amqp://guest:guest@localhost:5672//`；Redis（result backend）`redis://localhost:6379/0`。

5. 安装 PostgreSQL 17

```
# 禁用系统 PostgreSQL 模块，避免旧版覆盖
sudo dnf -qy module disable postgresql

# 官方 PGDG 源
sudo dnf -y install https://download.postgresql.org/pub/repos/yum/reporpms/EL-8-x86_64/pgdg-redhat-repo-latest.noarch.rpm

# 安装并初始化
sudo dnf -y install postgresql17 postgresql17-server postgresql17-contrib
sudo /usr/pgsql-17/bin/postgresql-17-setup initdb
sudo systemctl enable --now postgresql-17

# 设置 postgres 超级用户密码并创建项目库
sudo -u postgres psql -c "ALTER USER postgres WITH PASSWORD 'postgres';"
sudo -u postgres createdb messages -O postgres
```

说明：若已存在 `postgres` 角色，使用 `ALTER USER` 设置密码；避免重复 `CREATE USER` 报错。

6. 安装 Elasticsearch（建议 8.x）

```
# 导入 GPG key
sudo rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch

# 写入仓库文件（Here-Doc）
cat <<'EOF' | sudo tee /etc/yum.repos.d/elasticsearch.repo
[elasticsearch]
name=Elasticsearch repository for 8.x packages
baseurl=https://artifacts.elastic.co/packages/8.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md
EOF

# 安装与启动
sudo dnf -y install elasticsearch
sudo systemctl daemon-reload
sudo systemctl enable elasticsearch.service
sudo systemctl start elasticsearch.service
```

首次安装会生成 `elastic` 超级用户临时密码，可在日志中查看：

```
sudo grep "generated password for the elastic built-in superuser" /var/log/elasticsearch/elasticsearch.log
```

验证：

```
curl -u elastic:'<临时密码>' https://localhost:9200 -k
```

7. 依赖服务快速自检

```
sudo ss -lntp | egrep ':5672|:6379|:9200|:5432'
sudo systemctl status rabbitmq-server redis postgresql-17 elasticsearch --no-pager
```

2. 项目配置

1. 克隆与安装依赖

```
#作为windows文件中转到linux，压缩包解压一样的操作,可能要配置代理
git clone https://github.com/RioArisk/test.git
cd test
source ../venv/bin/activate    # 若不在同一目录，请调整路径
pip install -r requirements.txt
```

2. 环境变量（可选覆盖默认值）

```
export BROKER_URL=amqp://guest:guest@localhost:5672//
export RESULT_BACKEND=redis://localhost:6379/0
#注意这里的账号密码以及数据库表的名字根据实际填写，我的是postgres，postgres，表：
messages
export DB_URL=postgresql+psycopg2://postgres:postgres@localhost:5432/messages
#示例不带密码
export ES_URL=http://localhost:9200
#带密码
export ES_URL="http://elastic:Y+nZuoOR7_JMnaxy7jIY@localhost:9200"
```

3. 启动 Celery workers

```
fab workers --action=start
# 其他操作：
fab workers --action=restart
fab workers --action=stop
```

4. 处理数据

- 单个文件：

```
fab process-one --filename=/path/to/mail.eml
#示例
fab process-one --filename=/home/rioarisk/work/test/data/maildir/allen-
p/_sent_mail/1.
```

- 目录批量：

```
fab process --path=/path/to/maildir
#示例
fab process --path=/home/rioarisk/work/test/data/maildir
```

测试结果

```
(venv) [rioarisk@localhost test]$ fab process-one --
filename=/home/rioarisk/work/test/data/maildir/allen-p/_sent_mail/1.
Enqueued mail file for processing: /home/rioarisk/work/test/data/maildir/allen-
p/_sent_mail/1. (1153dfc9-07a9-4437-a743-fd51eac30d1f)
(venv) [rioarisk@localhost test]$ fab query-db --query="SELECT COUNT(*) FROM
messages;"
 count
-------
     1
(1 row)
```

注意事项

- 数据库“messages”需预先创建；程序仅会在已存在的数据库中自动建表 `messages`。建库示例：

  ```
  sudo -u postgres psql -c "CREATE DATABASE messages OWNER postgres;"
  ```

- 任务是异步执行，入队后需等待 workers 处理完成再查询；可查看 `celery-logs/` 下日志以确认执行状态。
- Fabric 的 `query-db`/`purge` 使用 `psql`，请确保 `DB_URL` 为 `postgresql://...`（不要包含 `+psycopg2` 驱动后缀）；应用自身可使用 `postgresql+psycopg2://...`。
- 文件扩展名不作限制，诸如 `1.` 的文件能被正常解析（基于内容解析，而非后缀）。
- 确保 RabbitMQ、Redis、PostgreSQL、Elasticsearch 服务已启动且网络可达。

5. 查看结果

- PostgreSQL：

```
fab query-db --query="SELECT COUNT(*) FROM messages;"
```

- Elasticsearch：

```
fab query-es --query="subject:meeting"
```

6. 清理

```
fab purge
```

7. 变更说明（与旧版差异）

- Python 升级到 3.9+；源码已移除 Python2 `print` 语法。
- Fabric 从 1.x 迁移至 2.5.0：
  - 使用 `@task` 装饰器与 `Context` 对象，命令示例改为 `fab task --arg=value`。
  - `local()` 改为 `c.run()`。
- 数据库从 MySQL 切换为 PostgreSQL 17：
  - 连接串通过 `DB_URL` 配置，默认 `postgresql+psycopg2://postgres:postgres@localhost:5432/messages`。
  - Fabric 中的数据库查询与清理命令改用 `psql`。
- Celery 兼容 Celery 5：
  - 配置项使用 `result_expires` 等键；支持通过环境变量覆盖 broker/backend。
- Elasticsearch 写入兼容 7/8：
  - 优先使用 `document=` 参数，若不支持则回退为 `body=`。

8. 运行验证

1. 启动依赖（RabbitMQ、Redis、PostgreSQL、Elasticsearch）
2. 安装依赖并设置环境变量
3. 启动 workers：`fab workers --action=start`
4. 处理一个示例邮件文件：`fab process-one --filename=./sample.eml`
5. 校验：
   - `fab query-db --query="SELECT COUNT(*) FROM messages;"`
   - `fab query-es --query="*:*"`

若遇到问题，请检查：

- 端口与凭据是否正确（BROKER_URL、RESULT_BACKEND、DB_URL、ES_URL）。
- PostgreSQL 权限与表是否已自动创建。