解决 Docker 中端口频繁占用及退出`docker compose logs -f`无法退出的问题

问题 1：docker compose logs -f 用 Ctrl+C 无法退出的解决

如果 `Ctrl+C` 无效，直接关闭当前终端窗口（或标签页）即可，这是最快捷的方式。

重新打开终端后，执行 `docker compose down` 可正常停止服务，不会影响容器状态。

问题 2：端口频繁占用的深层原因及解决

从多次端口被占用来看，核心问题是 **使用 `host` 网络时，容器直接占用宿主机端口，而系统中已有大量程序占用了常见端口**。解决方法是让容器使用独立网络，避免直接占用宿主机端口：

步骤 1：恢复容器网络（不使用 host 网络）

```
# 进入配置目录
cd /mnt/sda1/immich/config
# 恢复为默认桥接网络的 docker-compose.yml
cat << 'EOF' > docker-compose.yml
version: '3.8'
services:
  immich-server:
    container_name: immich_server
    image: ghcr.io/immich-app/immich-server:release
    ports:
      - "2283:2283"  # 宿主机端口:容器端口
    volumes:
      - ${UPLOAD_LOCATION}:/usr/src/app/upload
    env_file:
      - .env
    depends_on:
      - redis
      - database
    restart: always
  immich-machine-learning:
```

```
  container_name: immich_machine_learning
  image: ghcr.io/immich-app/immich-machine-learning:release
  volumes:
    - ${UPLOAD_LOCATION}:/usr/src/app/upload
    - model-cache:/cache
  env_file:
    - .env
  restart: always
 database:
  container_name: immich_postgres
  image: ghcr.io/immich-app/postgres:14-vectorchord0.4.3-pgvectors0.2.0@sha256:bcf63357
191b76a916ae5eb93464d65c07511da41e3bf7a8416db519b40b1c23
  environment:
    POSTGRES_PASSWORD: ${DB_PASSWORD}
    POSTGRES_USER: ${DB_USERNAME}
    POSTGRES_DB: ${DB_DATABASE_NAME}
  volumes:
    - ${DB_DATA_LOCATION}:/var/lib/postgresql/data
  restart: always
 redis:
  container_name: immich_redis
  image: docker.io/valkey/valkey:8@sha256:81db6d39e1bba3b3ff32bd3a1b19a6d69690f94a3
954ec131277b9a26b95b3aa
  volumes:
    - ${REDIS_DATA_LOCATION}:/data
  restart: always
volumes:
 model-cache:
EOF
```

步骤 2：恢复服务名连接（不用 IP）

```
# 修改 .env 用服务名连接
sed -i "s|DB_HOSTNAME=.*|DB_HOSTNAME=immich_postgres|g" .env
sed -i "s|REDIS_HOSTNAME=.*|REDIS_HOSTNAME=immich_redis|g" .env
# 恢复默认端口（2283，通过桥接网络映射，不会冲突）
sed -i "s|IMMICH_PORT=.*|IMMICH_PORT=2283|g" .env
```

步骤 3：重启服务

```
# 强制删除旧容器和网络
docker rm -f immich_server immich_postgres immich_redis immich_machine_learning
docker network rm immich_default
# 启动新配置
docker compose up -d
```

验证

```
# 查看日志
docker compose logs -f immich-server
```

此时容器使用独立的桥接网络，`2283` 端口通过容器映射到宿主机，不会与宿主机程序冲突。服务间通过容器名（`immich_postgres`、`immich_redis`）正常通信，最终日志会显示 `Server is running on port 2283`，访问 http://服务器IP:2283 即可使用。

这种方式从根本上解决了端口冲突和网络隔离问题，适合你的系统环境。

> （注：文档部分内容可能由 AI 生成）