StealthDB的运行环境需求主要包括：

Intel SGX支持的CPU。

Intel SGX PSW和SDK安装在/opt目录。

Intel SGX驱动版本2.0。

安装NASM汇编器。

3. 克隆StealthDB项目

使用Git克隆StealthDB项目到你的本地机器。

git clone https://github.com/cryptograph/stealthdb.git

4. 使用Docker构建和运行StealthDB

按照项目README中提供的指南，使用Docker构建和运行StealthDB。

打开命令行界面。

导航到克隆的StealthDB项目目录。

根据项目中的指南构建Docker镜像：

docker build -f docker/Dockerfile -t stealthdb:1.0 docker/

5.运行Docker容器：

docker run -it -d --rm --device=/dev/isgx --volume=/var/run/aesmd/aesm.socket:/var/run/aesmd/aesm.socket -p 5432:5432 --name sdb stealthdb:1.0

2. 安装Windows上的Intel SGX驱动和软件

由于你使用的是Windows 11，你需要在Windows环境中安装Intel SGX驱动和平台软件（PSW）。以下是安装步骤：

下载SGX驱动和PSW：访问Intel官方网站，下载适用于Windows 11的Intel SGX驱动和平台软件（Platform Software, PSW）。

安装SGX驱动：下载完成后，按照安装程序指示安装SGX驱动。

安装PSW：安装驱动后，继续安装平台软件。这个软件包含运行SGX应用程序所需的一些关键组件。

确认/dev/isgx设备：安装完成后，确保设备管理器中出现了/dev/isgx设备。这表明SGX驱动已正确安装。

3. 安装Docker并配置环境

使用Docker在Windows 11上模拟Ubuntu环境来开发和运行SGX应用程序：

安装Docker for Windows：从Docker官方网站下载并安装Docker Desktop for Windows。

配置Docker：安装完成后，可能需要在Docker设置中启用某些功能，如Hyper-V。

拉取Ubuntu镜像：使用Docker命令行拉取Ubuntu 16.04镜像：

docker pull ubuntu:16.04

4.运行Ubuntu容器：使用Docker运行Ubuntu容器，并将必要的SGX设备和目录映射到容器中：

docker run -v /path/to/sgx-application:/sgx-app --device /dev/isgx -it ubuntu:16.04

将/path/to/sgx-application替换为你的SGX应用程序的路径。

定位你的SGX应用程序路径

SGX项目位置：这个路径应该是你在Windows 11系统上开发的SGX应用程序的文件夹。例如，如果你在D:\Projects\MySGXApp文件夹中保存了你的SGX项目，那么这就是你需要使用的路径。

映射到Docker容器：当你运行Docker容器时，你会希望将这个项目文件夹映射（或挂载）到容器内部。这样做可以使容器内的环境访问和使用这些文件。

示例

假设你的SGX项目位于D:\Projects\MySGXApp，那么你可以使用以下命令来运行Docker容器，并将该项目文件夹映射到容器内：

docker run -v D:/Projects/MySGXApp:/sgx-app --device /dev/isgx -it ubuntu:16.04

这里的-v D:/Projects/MySGXApp:/sgx-app部分告诉Docker将你的Windows路径D:/Projects/MySGXApp映射到容器内的/sgx-app路径。在容器内部，你可以在/sgx-app找到你的SGX项目文件。

环境搭建

安装 Docker:

确保您的 Windows 11 系统上已经安装了 Docker Desktop。如果尚未安装，请访问 Docker 官网 进行下载和安装。

克隆 StealthDB 项目:

打开命令行工具（如CMD或PowerShell）。

克隆项目仓库：

git clone https://github.com/cryptograph/stealthdb.git

git clone https://github.com/xsn1210/stealthdb.git

进入克隆的项目目录：

cd stealthdb

构建 Docker 镜像:

在项目目录中，使用 Makefile 提供的 docker 命令来构建 Docker 镜像：

make docker

这个命令会执行两个操作：

构建名为 stealthdb:1.0 的 Docker 镜像。

运行一个名为 sdb 的 Docker 容器，容器内部运行 StealthDB。

运行 StealthDB 项目

检查 Docker 容器状态:

确认 Docker 容器已成功运行

docker ps

查找名为 sdb 的容器，确认其状态为运行中。

连接到 StealthDB 数据库:

使用任何 PostgreSQL 兼容的客户端工具连接到运行中的容器。由于容器将 PostgreSQL 端口 5432 映射到宿主机的同一端口，您可以使用本地主机地址和端口 5432 进行连接。

运行 SQL 查询:

连接到数据库后，您可以开始运行 SQL 查询。

按照项目文档中的指南，您可以尝试创建加密扩展、生成和加载密钥，以及执行加密查询。例如：

CREATE EXTENSION encdb;

SELECT generate\_key();

SELECT load\_key(0);

SELECT pg\_enc\_int4\_encrypt(1) + pg\_enc\_int4\_encrypt(2);

SELECT pg\_enc\_int4\_decrypt(pg\_enc\_int4\_encrypt(1) + pg\_enc\_int4\_encrypt(2));

2. 安装 pgTAP 工具

pgTAP 是一个 PostgreSQL 数据库的测试框架，它允许您编写和运行测试用例。要在 Docker 容器中安装 pgTAP，您需要执行以下步骤：

进入运行中的容器：

docker exec -it sdb bash

这将打开一个 shell，让您可以在容器内运行命令。

在容器内安装 pgTAP：

apt-get update

apt-get install libtap-parser-sourcehandler-pgtap-perl

这将在容器内安装 pgTAP 的依赖。

3. 运行测试

在容器的 shell 中，运行测试：

pg\_prove -U postgres -d postgres encdb/run\_test.sql

这将使用 pgTAP 运行 encdb 目录下的 SQL 测试脚本。

观察测试输出。pgTAP 会提供每个测试用例的结果。

4. 断开容器

完成测试后，您可以输入 exit 来离开容器的 shell。

5. 使用 StealthDB

运行测试之外，您还可以使用 PostgreSQL 客户端工具连接到容器中的数据库，以验证 StealthDB 功能。例如，您可以尝试以下操作：

连接到数据库：

使用您选择的 PostgreSQL 客户端工具连接到 localhost 的 5432 端口。

创建加密扩展和执行加密查询：

CREATE EXTENSION encdb;

SELECT generate\_key();

SELECT load\_key(0);

SELECT pg\_enc\_int4\_encrypt(1) + pg\_enc\_int4\_encrypt(2);

SELECT pg\_enc\_int4\_decrypt(pg\_enc\_int4\_encrypt(1) + pg\_enc\_int4\_encrypt(2));

@powershell -NoProfile -ExecutionPolicy unrestricted -Command "iex ((new-object net.webclient).DownloadString('http://bit.ly/psChocInstall'))" && SET PATH=%PATH%;%systemdrive%\chocolatey\bin

@"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -ExecutionPolicy Bypass -Command "iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"

choco install make

# Builds Intel SGX libraries and binaries required by StealthDB.

# Does not include those corresponding to the AESMD service and SGX driver.

SGX\_MODE ?= HW

export SGX\_MODE

USE\_OPT\_LIBS = 1

export USE\_OPT\_LIBS

# SDK\_REV := sgx\_2.0 # 注释掉或移除这行

SDK\_SRC := linux-sgx # 修改这里

SDK\_BUILD\_DIR := $(SDK\_SRC)/build/linux

SDK\_INSTALL\_PATH := /opt/intel/sgxsdk

SGX\_BUILD\_LIBS := libsgx\_trts.a libsgx\_tservice.a libsgx\_tstdc.a libsgx\_tcrypto.a libsgx\_tstdcxx.a libsgx\_tcxx.a

SGX\_RUNTIME\_LIBS := libsgx\_urts.so libsgx\_uae\_service.so

SGX\_LIBS\_INSTALL\_DIR := $(SDK\_INSTALL\_PATH)/lib64

SGX\_BINS := sgx\_sign sgx\_edger8r

SGX\_BINS\_INSTALL\_DIR := $(SDK\_INSTALL\_PATH)/bin/x64

SGX\_HEADERS\_INSTALL\_DIR := $(SDK\_INSTALL\_PATH)/include

.PHONY: all

all: $(addprefix $(SGX\_LIBS\_INSTALL\_DIR)/, $(SGX\_BUILD\_LIBS) $(SGX\_RUNTIME\_LIBS)) $(addprefix $(SGX\_BINS\_INSTALL\_DIR)/, $(SGX\_BINS)) sgx\_headers

$(SGX\_LIBS\_INSTALL\_DIR)/libsgx\_%: $(SDK\_BUILD\_DIR)/libsgx\_% | $(SGX\_LIBS\_INSTALL\_DIR)

cp $(SDK\_BUILD\_DIR)/libsgx\_$\* $@

$(SDK\_BUILD\_DIR)/libsgx\_%.a: | $(SDK\_SRC)

$(MAKE) -C $(SDK\_SRC)/sdk $\*

$(SDK\_BUILD\_DIR)/libsgx\_%.so: | $(SDK\_SRC)

$(MAKE) -C $(SDK\_SRC)/psw CXXFLAGS="-Wno-unused-parameter -fPIC" $\*

$(SGX\_BINS\_INSTALL\_DIR)/sgx\_%: $(SDK\_BUILD\_DIR)/sgx\_% | $(SGX\_BINS\_INSTALL\_DIR)

cp $(SDK\_BUILD\_DIR)/sgx\_$\* $@

$(SDK\_BUILD\_DIR)/sgx\_sign: | $(SDK\_SRC)

$(MAKE) -C $(SDK\_SRC)/sdk signtool

$(SDK\_BUILD\_DIR)/sgx\_edger8r: | $(SDK\_SRC)

$(MAKE) -C $(SDK\_SRC)/sdk edger8r

.PHONY: sgx\_headers

sgx\_headers: | $(SGX\_HEADERS\_INSTALL\_DIR)

cp -R $(SDK\_SRC)/common/inc/\* $|

$(SGX\_HEADERS\_INSTALL\_DIR) $(SGX\_BINS\_INSTALL\_DIR) $(SGX\_LIBS\_INSTALL\_DIR):

mkdir -p $@

$(SDK\_SRC): # 修改这里

git clone --depth 1 https://github.com/xsn1210/linux-sgx11.git $(SDK\_SRC)

.PHONY: uninstall

uninstall:

$(RM) $(addprefix $(SGX\_LIBS\_INSTALL\_DIR), $(SGX\_LIBS))\

$(addprefix $(SGX\_BINS\_INSTALL\_DIR), $(SGX\_BINS))\

-r $(SGX\_HEADERS\_INSTALL\_DIR)

.PHONY: clean

clean:

$(RM) -r $(SDK\_SRC)