高级程序语言设计-Ⅱ代码实践的环境说明文档

Introduction

本次代码环境配置方案采取WSL+docker/原生linux+docker方案进行实践环境配置，力求还原真实C++代码开发的流程与情形。

编辑器/IDE推荐：VScode/Clion/Vim

为什么使用Linux+Docker

问一问万能的的ChatGPT

13 为什么要在Linux环境下学习C++

在Linux环境下学习C++有以下几个好处：

1. C++在Linux系统中的应用广泛：Linux是一个开放源代码的操作系统，许多开源软件、服务器和设备都是使用C++编写的。学习C++在Linux系统中的应用可以帮助您更深入地了解和理解Linux系统。
2. 开发工具支持：Linux上有许多强大的开发工具，如GNU编译器、Make工具、GDB调试器等，它们对于C++开发非常友好。通过在Linux环境下学习C++，您可以更加深入地了解这些工具的使用，提高开发效率。
3. 系统编程：Linux系统编程是C++开发的重要领域之一。学习C++在Linux环境下的编程可以帮助您更好地了解系统编程的相关概念和技术，如进程管理、内存管理、文件操作等。
4. 开源社区：Linux社区是一个庞大的开源社区，拥有数以万计的开源项目。在Linux环境下学习C++可以帮助您更好地参与到开源社区中，学习和贡献更多的开源项目。
5. 跨平台开发：C++是一种跨平台语言，可以在Windows、Linux、Mac OS X等操作系统中运行。在Linux环境下学习C++可以帮助您更好地了解C++在不同操作系统中的应用和开发。

13 docker是什么

Docker 是一个开源的容器化平台，它可以帮助开发者将应用程序及其依赖打包到一个可移植的容器中，以便在不同的环境中运行。Docker 容器可以在几乎任何操作系统上运行，因为它们是轻量级的、独立的软件包，它们包含应用程序、运行时环境、系统工具、库和设置等内容。这使得应用程序可以在不同的计算机上、不同的操作系统上或不同的云平台上无缝地运行。

使用 Docker，开发者可以更快地构建、测试和部署应用程序。Docker 提供了一个简单的、一致的开发环境，可以减少在不同操作系统和服务器上运行应用程序时出现的不兼容性问题。此外，Docker 还支持自动化部署和容器编排，使得开发者可以更轻松地扩展应用程序的规模。

13 使用docker有什么好处

使用Docker有以下好处：

1. 隔离性：Docker容器提供了隔离的环境，避免了应用程序之间的冲突，也可以保证应用程序的稳定性。
2. 可移植性：Docker容器可以在任何平台上运行，包括云、物理服务器、虚拟机等。
3. 快速部署：Docker容器可以快速地创建和部署，可以避免复杂的配置和安装流程。
4. 资源利用率：Docker容器可以共享操作系统内核，因此它们相对于虚拟机来说更轻量级，更节省资源。
5. 版本控制：Docker容器可以使用版本控制来管理应用程序的变更和升级，方便进行回滚和管理。
6. 微服务架构：Docker容器适合构建微服务架构，可以将应用程序拆分成多个小的、独立的容器来管理和维护。

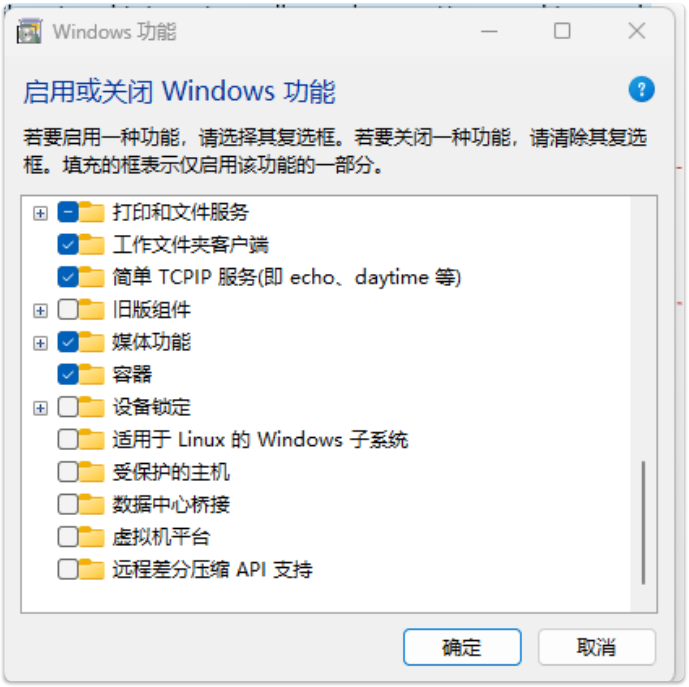
Windows下安装docker与配置（WSL+Docker Desktop）

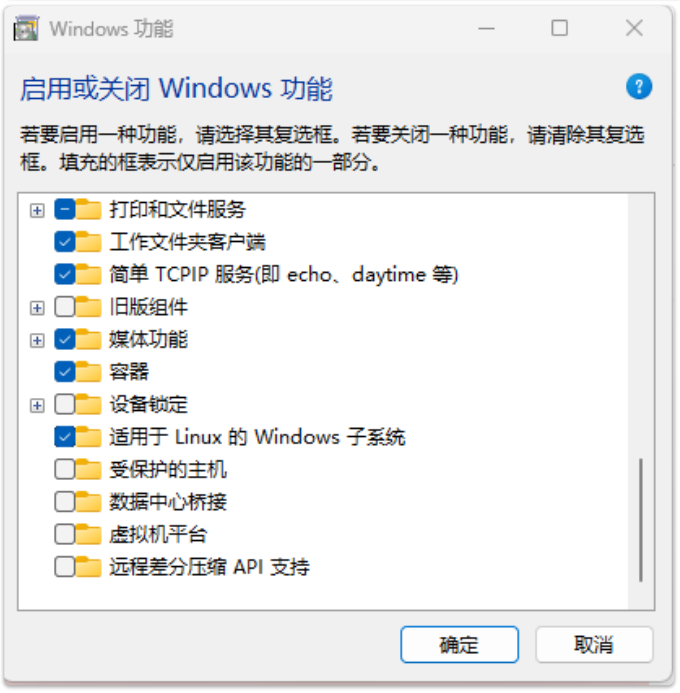
这里我以我的windows台式机作为例子，参考https://learn.microsoft.com/zh-cn/windows/wsl/install
进行安装
`win+r`在框中输入`winver`查看当前的版本，必须运行 Windows 10 版本 2004 及更高版本（内部版本 19041 及更高版

本）或 Windows 11 才能使用命令安装。

打开开发者选项

启用WSL





安装WSL命令

在管理员模式下打开 PowerShell 或 Windows 命令提示符，方法是右键单击并选择“以管理员身份运行”，输入 wsl --install 命令

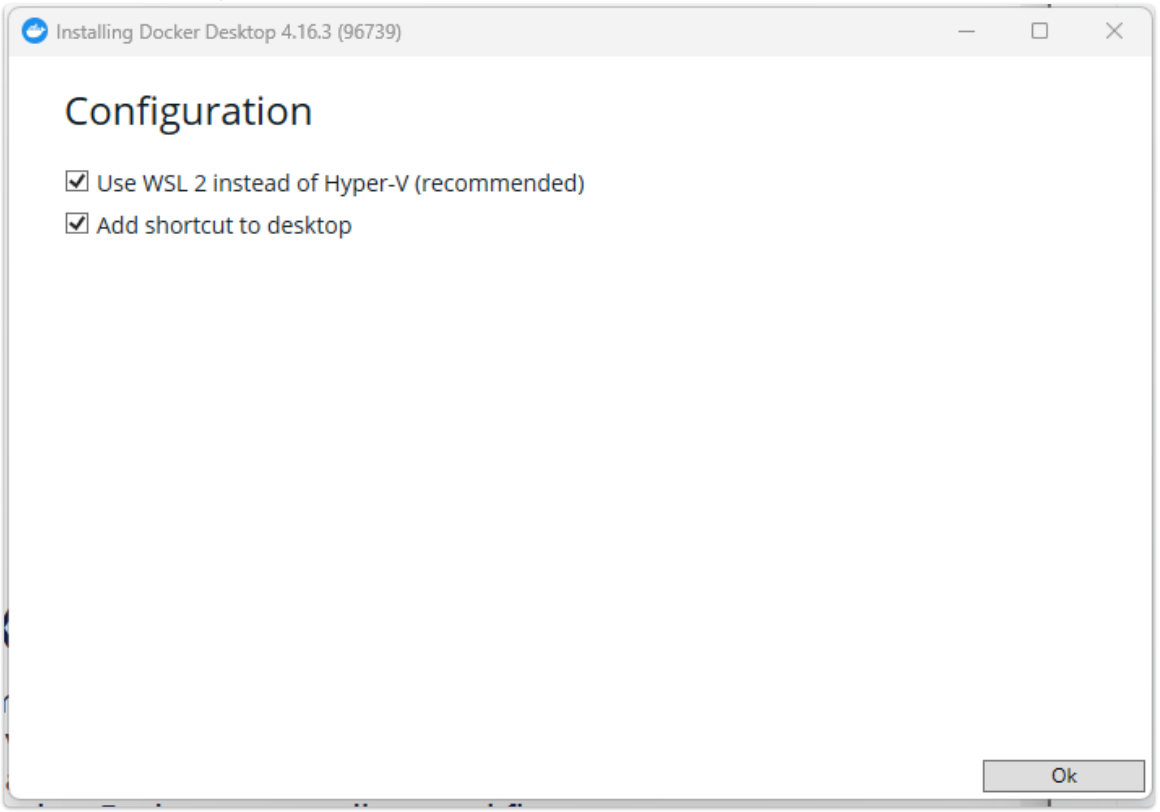
```
wsl --install
```

```
Microsoft Windows [版本 10.0.22621.1265]
(c) Microsoft Corporation。保留所有权利。

C:\Users\YuCDg>wsl --install
适用于 Linux 的 Windows 子系统已安装。

以下是可安装的有效分发的列表。
请使用“wsl --install -d <分发>”安装。

NAME                                   FRIENDLY NAME
Ubuntu                                 Ubuntu
Debian                                 Debian GNU/Linux
kali-linux                             Kali Linux Rolling
Ubuntu-18.04                           Ubuntu 18.04 LTS
Ubuntu-20.04                           Ubuntu 20.04 LTS
Ubuntu-22.04                           Ubuntu 22.04 LTS
OracleLinux_8_5                        Oracle Linux 8.5
OracleLinux_7_9                        Oracle Linux 7.9
SUSE-Linux-Enterprise-Server-15-SP4    SUSE Linux Enterprise Server 15 SP4
openSUSE-Leap-15.4                     openSUSE Leap 15.4
openSUSE-Tumbleweed                    openSUSE Tumbleweed

C:\Users\YuCDg>
```

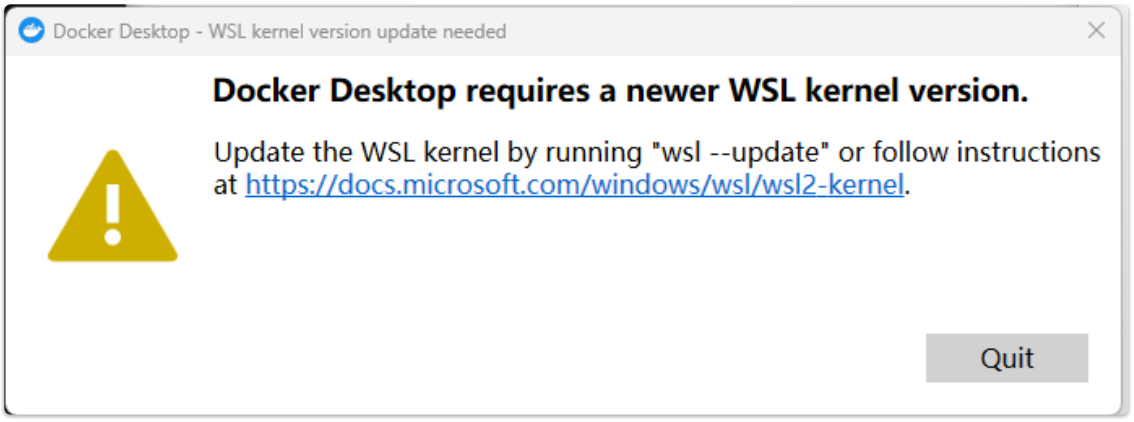
说明已经装好wsl所需要的底层环境了，这里我们先不急着装相应的发行版。

Docker Desktop安装

进入页面https://www.docker.com/products/docker-desktop/
下载docker desktop for windows，点击安装，安装完毕后重启



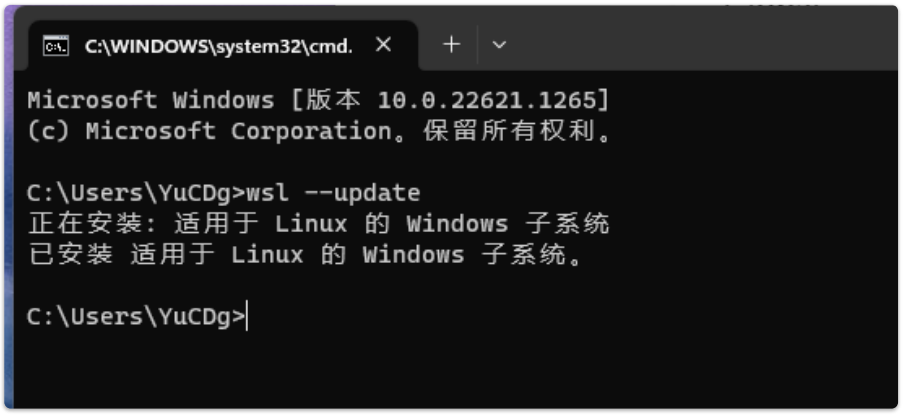
打开，如果出现



在管理员模式下打开 PowerShell 或 Windows 命令提示符，输入

```
wsl --update
```

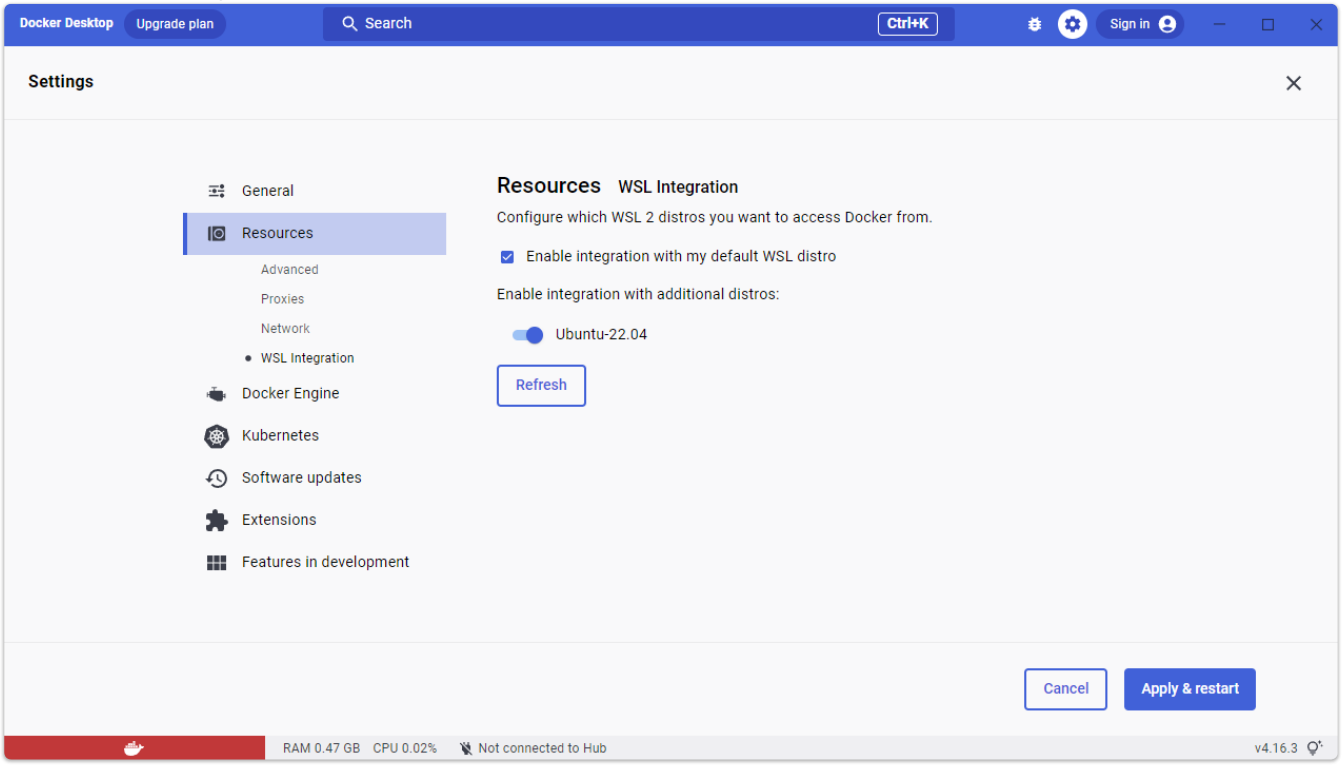
等待如下页面：



然后输入

```
wsl --shutdown
```

重启docker desktop

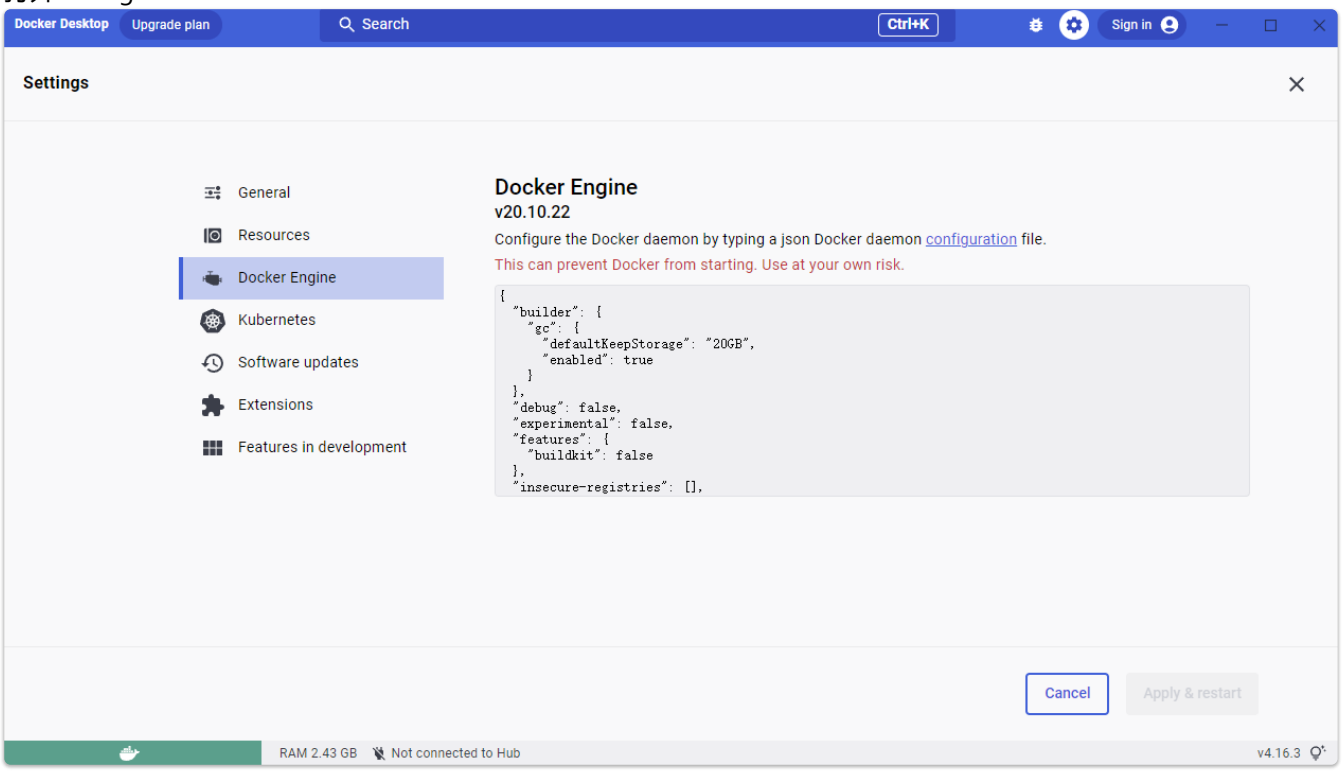


接下来是docker的网络配置，如果是已经挂了vpn的可以跳过这一步，如果没有挂我们进行换源

我么先使用https://github.com/anjia0532/gcr.io_mirror

查询对应的最快镜像站，这里我们使用ustc和163源

打开settings



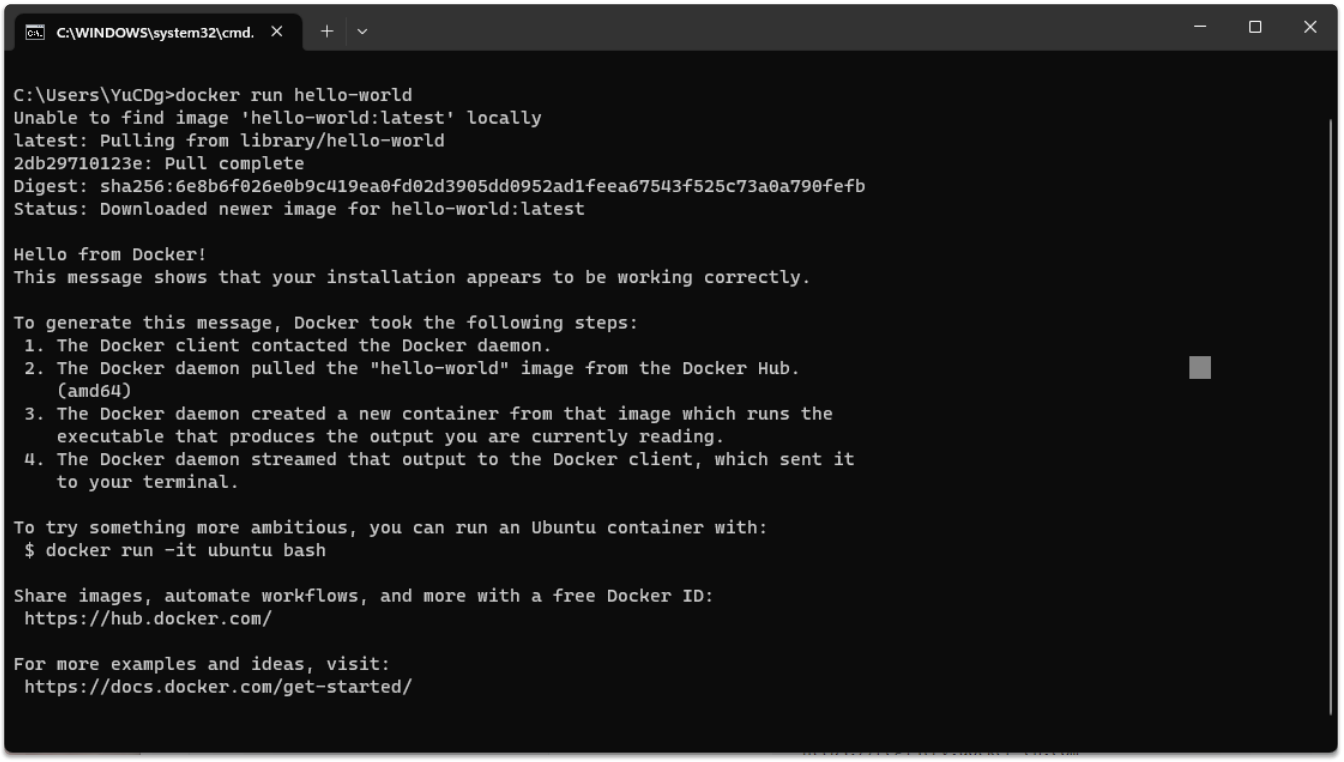
替换成下面的文字

```
{
  "builder": {
    "gc": {
      "defaultKeepStorage": "20GB",
      "enabled": true
    }
  },
  "debug": false,
  "experimental": false,
  "features": {
    "buildkit": false
  },
  "insecure-registries": [],
  "registry-mirrors": [
  https://docker.mirrors.ustc.edu.cn/",
  https://hub-mirror.c.163.com",
  https://registry.docker-cn.com"
  ]
}
```

换源完成之后重启，打开终端

```
docker run hello-world
```

见到如下即是安装完成



创建Docker容器

首先我们要下载镜像，然后创建容器。（容器就是我们最终需要的！）
如果你有梯子，要自己下载镜像，就用下面这种方法

使用dockerfile生成镜像

使用我们提供的dockerfile文件，cd到对应目录，我这里以桌面举例（之后环境更改后使用的dockerfile操作同理）

```
docker build -t scucpphw_test_img .
```

```
C:\Users\YuCDg\Desktop>docker build -t scucpphw_test_img .
Sending build context to Docker daemon  9.216kB
Step 1/12 : FROM gcc:11.2.0
11.2.0: Pulling from library/gcc
6aefca2dc61d: Pull complete
967757d56527: Pull complete
c357e2c68cb3: Pull complete
c766e27afb21: Pull complete
32a180f5cf85: Pull complete
769273b4685a: Pull complete
b17d9f857209: Pull complete
1b75655eab73: Pull complete
56fff7f084d0: Pull complete
```

经过下载之后，可能要等比较久

```
docker images
```

```
C:\Users\YuCDg\Desktop>docker images
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
scucpphw_test_img   latest    8db19d3ac35b   2 minutes ago   1.3GB
alpine              3.16.3    bfe296a52501   3 months ago    5.54MB
gcc                 11.2.0    c2968a627869   10 months ago   1.23GB
hello-world         latest    feb5d9fea6a5   17 months ago   13.3kB
```

可以看到对应的镜像，说明我们安装成功了对应的docker环境。

创建容器，并进行目录映射

`-v $(你的本地工作区):/ws`，就是将本地工作区映射到最终环境下的/ws，类似于共享文件夹。首先你需要在windows内创建一个合适的文件夹，本节课所有的实验都是在这个文件夹内进行，这个文件夹将会跟最终容器的/ws同步，这样的好处是在删除容器后，写的代码不会丢掉

```
docker run -it -d -v $(你的本地工作区):/ws scucpphw_test_img:latest /bin/bash
```

这里我还是用桌面举例

```
 docker run -i -d -v C:\Users\YuCDg\Desktop\SCUCPP:/ws scucpphw_test_img:latest /bin/bash
```

输出这样的

```
PS C:\Users\YuCDg\Desktop> docker run -i -d -v C:\Users\YuCDg\Desktop\SCUCPP:/ws
scucpphw_test_img:latest /bin/bash
f9cc010728d26936f3c83ba6d256414ddffe3108ce3ee34c7c06139b18d67c59
PS C:\Users\YuCDg\Desktop> docker ps
CONTAINER ID   IMAGE                      COMMAND       CREATED          STATUS          PORTS
NAMES
f9cc010728d2   scucpphw_test_img:latest   "/bin/bash"   37 seconds ago   Up 36 seconds   22/tcp
vigilant_knuth
PS C:\Users\YuCDg\Desktop> docker exec -it f9cc010728d2 /bin/bash
```

然后

```
docker ps
docker exec -it f9cc010728d2 /bin/bash
```

docker exec -it <container id（我这里是f9cc010728d2）> /bin/bash
上面的f9cc010728d2用你自己的container id代替
这样我们就进入了这个容器的bash，在bash中输入以下指令安装gdb与vim

安装gdb与Vim

```
apt-get update
apt-get upgrade
apt-get install gdb vim
```

```
PS C:\Users\YuCDg\Desktop> docker run -i -d -v C:\Users\YuCDg\Desktop\SCUCPP:/ws scucpphw_test_img:latest /bin/bash
f9cc010728d26936f3c83ba6d256414ddffe3108ce3ee34c7c06139b18d67c59
PS C:\Users\YuCDg\Desktop> docker ps
CONTAINER ID   IMAGE                      COMMAND       CREATED          STATUS          PORTS     NAMES
f9cc010728d2   scucpphw_test_img:latest   "/bin/bash"   37 seconds ago   Up 36 seconds   22/tcp    vigilant_knuth
PS C:\Users\YuCDg\Desktop> docker exec -it f9cc010728d2 /bin/bash
root@f9cc010728d2:/# pwd
/
root@f9cc010728d2:/# cd /ws
```

导入实验代码

先讲一下如何启动对应的docker容器

```
docker ps
docker exec -it f9cc010728d2 /bin/bash
```

docker exec -it <container id（我这里是f9cc010728d2） > /bin/bash
上面的f9cc010728d2用你自己的container id代替
进入之后

```
cd ws
```

在windows内把code复制进入ws即可

Linux下安装docker（Ubuntu为例）

如果你用windows，你就不用Linux下配置

命令安装docker

参考https://docs.docker.com/engine/install/ubuntu/

```
sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

sudo docker run hello-world

sudo docker run hello-world
```

```
(base) yuchuan@yuchuan:~$ sudo docker run hello-world
[sudo] password for yuchuan:

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

导入dockerfile

进入下载好dockerfile的目录

```
docker build -f Dockerfile -t scucpphw_test_img .

docker images
```

显示scucpphw_test_img即为导入成功

```
(base) yuchuan@yuchuan:~$ docker images
REPOSITORY                TAG         IMAGE ID       CREATED         SIZE
scucpphw_test_img         latest      5a464f4dec80   8 minutes ago   1.3GB
nvcr.io/nvidia/pytorch    23.01-py3   9eda6061497d   4 weeks ago     20.5GB
gcc                       11.2.0      c2968a627869   10 months ago   1.23GB
hello-world               latest      feb5d9fea6a5   17 months ago   13.3kB
```

创建docker容器并映射

```
docker ps
docker run -it -d -v /home/yuchuan/SCUCPP:/ws scucpphw_test_img:latest /bin/bash


docker exec -it 99b93e33383a /bin/bash
```

```
(base) yuchuan@yuchuan:~$ docker ps
CONTAINER ID   IMAGE                      COMMAND       CREATED         STATUS         PORTS     NAMES
99b93e33383a   scucpphw_test_img:latest   "/bin/bash"   4 minutes ago   Up 4 minutes   22/tcp    determined_shaw
```

```
(base) yuchuan@yuchuan:~$ docker exec -it 99b93e33383a /bin/bash
root@99b93e33383a:/# g++ --version
g++ (GCC) 11.2.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

root@99b93e33383a:/# apt update
Get:1 http://security.debian.org/debian-security bullseye-security InRelease [48.4 kB]
Get:2 http://deb.debian.org/debian bullseye InRelease [116 kB]
Get:3 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]
Get:4 http://security.debian.org/debian-security bullseye-security/main amd64 Packages [228 kB]
Get:5 http://deb.debian.org/debian bullseye/main amd64 Packages [8183 kB]
Get:6 http://deb.debian.org/debian bullseye-updates/main amd64 Packages [14.6 kB]
Fetched 8634 kB in 2s (5018 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
101 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@99b93e33383a:/# apt upgrade
```

安装GDB与Vim

```
apt update
apt upgrade
apt install gdb
apt install vim
```

导入实验代码

同windows，把代码复制进去就可以了

安装完成后的测试

启动docker，编辑hello_world.cpp

```
cd /ws/code/1_helloworld/
vim hello_world.cpp
```

替换如下

```
// ========================  C++ Way  ==============================
#include <iostream>

void hello_cpp() {
```

```
}

// ========================  C Way  ==============================

#include "stdio.h"
#include "stdlib.h"

void hello_c() {
    printf("%s", "Hello, world!\n");
}


// ========================  Assembly Way  ==============================

#include "stdio.h"
#include "stdlib.h"

/*
 * Wrapper function for convenience
 */
void myputs(char *s) {
    puts(s);
}

/*
 * This will probably not work on your computer.
 * Assembly is not at all portable; a good
 * reason to avoid using it!
 *
 * Those of you who have taken 107 should
 * be able to somewhat see what is happening here.
 */
int hello_as() {
    /* The assembly literally writes the hex representation
     * of as big a portion of the string as it can into the addresses
     * at range rsp to rsp + 0xd. That range is exactly 12 bytes long
     * as there are 12 characters in the "Hello, wordl!" string.
     */
    // 方法1
    asm("sub    $0x20,%rsp\n\t"
        "movabs $0x77202c6f6c6c6548,%rax\n\t"   // moves "Hello, w" portion to mem at $rsp
        "mov    %rax,(%rsp)\n\t"
        "movl   $0x646c726f, 0x8(%rsp)\n\t"     // moves "orld"  portion to mem at $rsp + 8
        "movw   $0x21, 0xc(%rsp)\n\t"           // moves "!" portion to mem at $rsp + 12
        "movb   $0x0,0xd(%rsp)\n\t"             // moves string null terminator to mem at $rsp + 13
        "leaq    (%rsp),%rax\n\t"               // loads address of $rsp as first argument to puts
        "mov    %rax,%rdi\n\t"
        "call  _Z6myputsPc\n\t"                // calls puts
        "add    $0x20, %rsp\n\t"
    );
    // 方法2
    char *s = "Hello, wrold!";
    asm("mov    $0x402012,%edi\n\t"
        "call  _Z6myputsPc\n\t"
        "mov    $0x0,%eax\n\t"
    );
    return EXIT_SUCCESS;
}


int main() {
```

```
    // hello_cpp();
    hello_c();
    hello_as();
}
```

保存后

```
g++ hello_world.cpp
./a.out
```

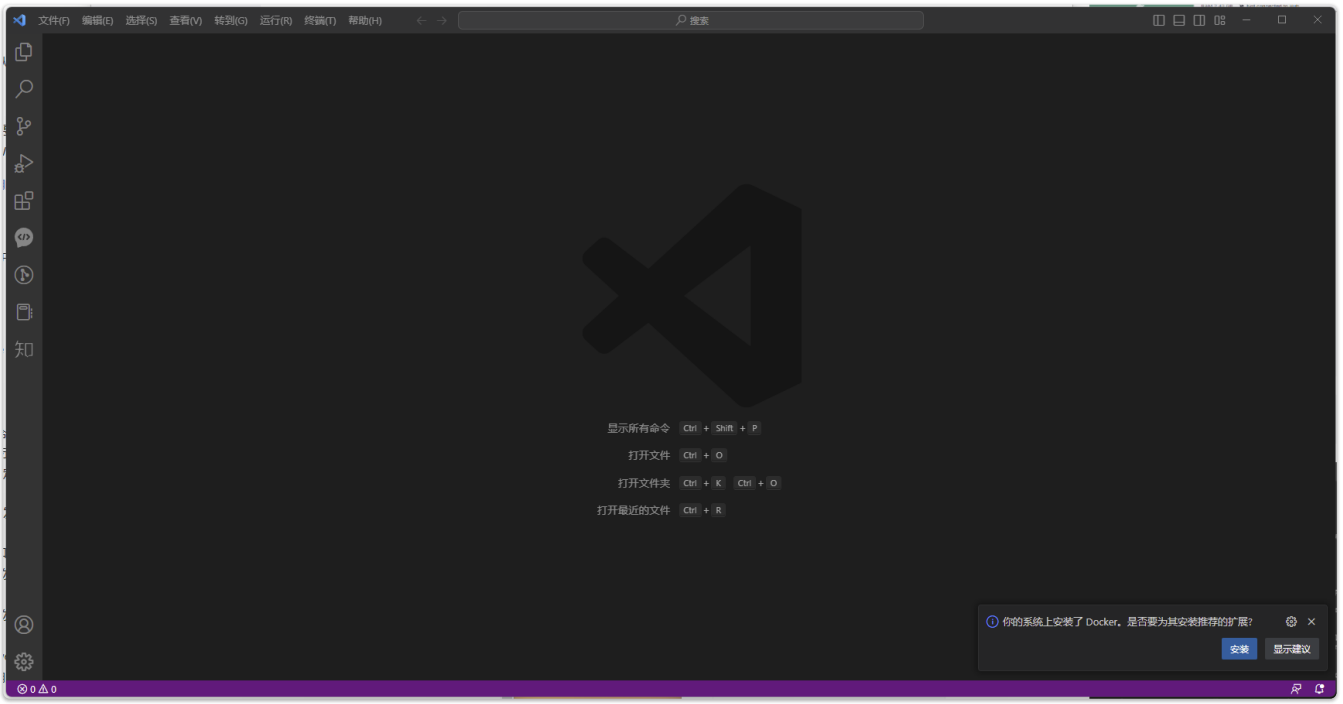
或者

```
./build.sh
```

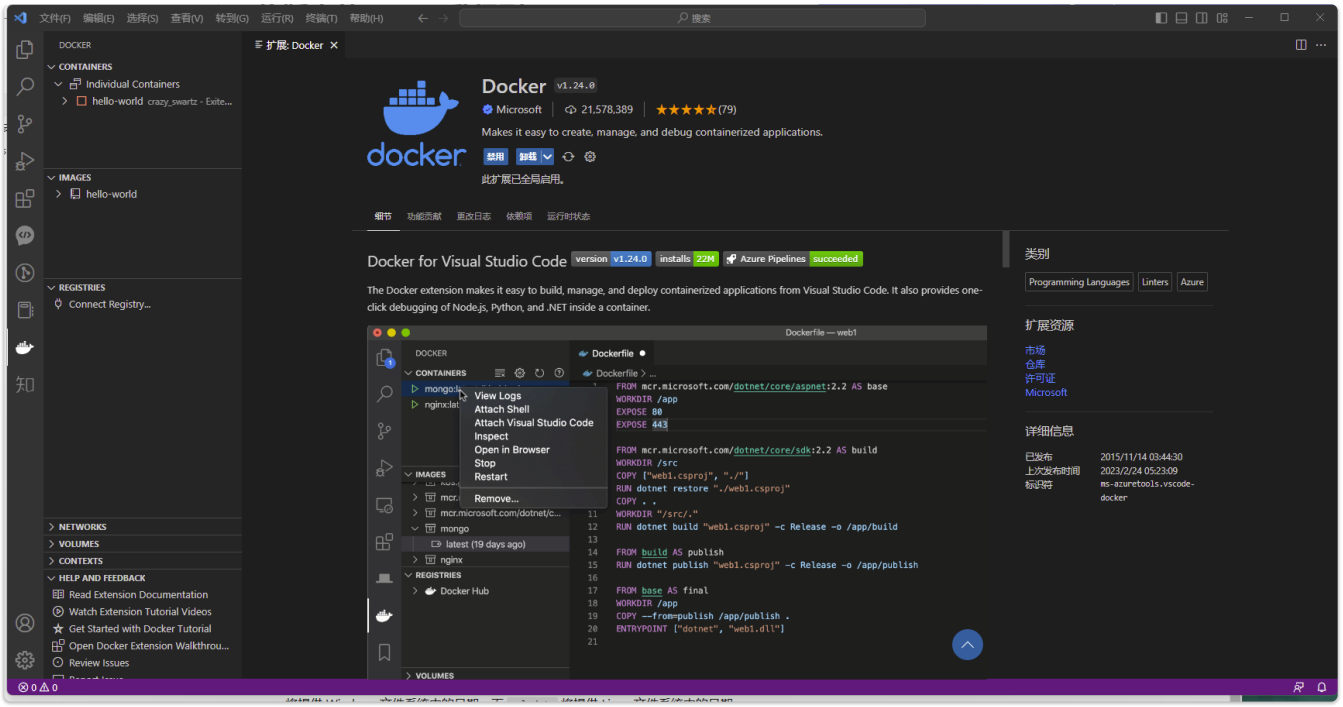
打印Hello, wrold!即为成功

VScode/Clion 配置

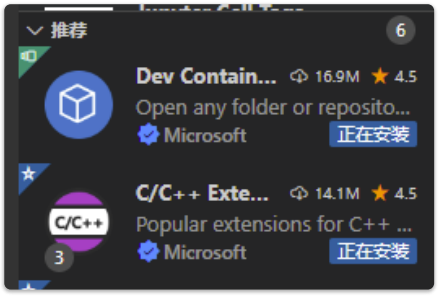
打开Vscode，如果你没安装过docker



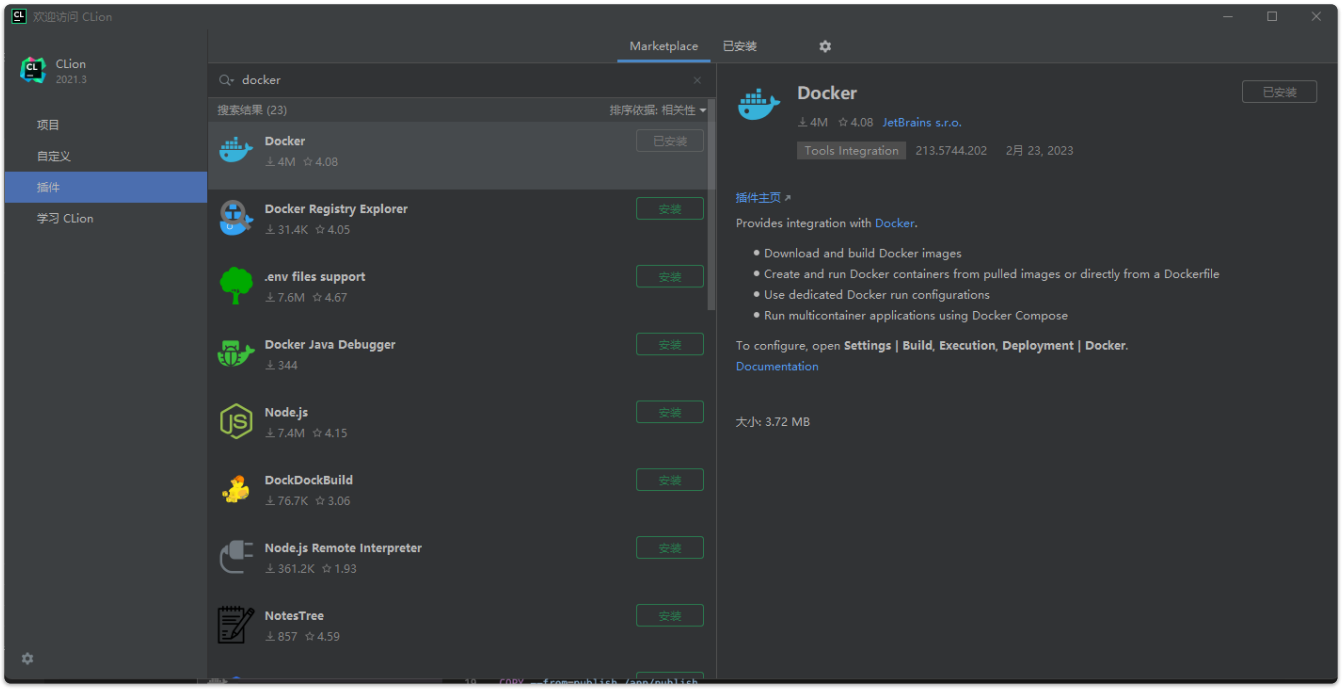
你可以看到一个自动提示，安装好后点击左侧可以看到



顺便安装



或者使用clion，打开下载相关插件



附录

Dockerfile

```
FROM gcc:11.2.0

RUN sed -i s@/archive.ubuntu.com/@/mirrors.aliyun.com/@g /etc/apt/sources.list\
```

```
    && apt-get clean\
    && apt-get -qq update \
    && apt-get -qq install --no-install-recommends openssh-server \
    && apt-get -qq install --no-install-recommends sudo \
    && apt-get -qq install --no-install-recommends cmake \
    && apt-get -qq install --no-install-recommends rsync \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*
# setup ssh for use ubuntu, password scucpp
RUN useradd -rm -d /home/ubuntu -s /bin/bash -g root -G sudo -u 1000 ubuntu
RUN  echo 'ubuntu:scucpp' | chpasswd
RUN service ssh start
EXPOSE 22
# install google test
WORKDIR /usr/src/libraries
RUN git clone --depth=1 -b main https://github.com/google/googletest.git
WORKDIR /usr/src/libraries/googletest/build
RUN cmake .. \
    && make \
    && make install
WORKDIR /
CMD ["./main"]
```