



**ASR582X 系列**

# **开发环境搭建指南**

文档版本 1.0.0

发布日期 2022-11-29

版权所有 © 2022 翱捷科技

## 关于本文档

本文档旨在指导用户快速入门，搭建 ASR582X 系列 Wi-Fi+BLE Combo SoC 芯片开发环境。

## 读者对象

本文档主要适用于以下工程师：

- 单板硬件开发工程师
- 软件工程师
- 技术支持工程师

## 产品名称

本文档适用于 ASR582X 系列 Wi-Fi+BLE Combo SoC 芯片。

## 版权公告

版权归 © 2022 翱捷科技股份有限公司所有。保留一切权利。未经翱捷科技股份有限公司的书面许可，不得以任何形式或手段复制、传播、转录、存储或翻译本文档的部分或所有内容。

## 商标声明



ASR、翱捷和其他翱捷商标均为翱捷科技股份有限公司的商标。

本文档提及的其他所有商标名称、商标和注册商标均属其各自所有人的财产，特此声明。

## 免责声明

翱捷科技股份有限公司对本文档内容不做任何形式的保证，并会对本文档内容或本文中介绍的产品进行不定期更新。

本文档仅作为使用指导，本文的所有内容不构成任何形式的担保。本文档中的信息如有变更，恕不另行通知。

本文档不负任何责任，包括使用本文档中的信息所产生的侵犯任何专有权行为的责任。

# 翱捷科技股份有限公司

地址：上海市浦东新区科苑路399号张江创新园10号楼9楼 邮编：201203

官网：<http://www.asrmicro.com/>

## 文档修订历史

日期	版本号	发布说明
2022.09	0.0.1	内部版本首次发布。
2022.09	0.0.2	增加 Docker 环境搭建。
2022.10	0.0.3	修改 Docker 镜像获取，使用 ASR 通用镜像。
2022.10	0.0.4	增加 Linux 下 make 验证。
2022.12	1.0.0	完善文档格式和布局。

# 目录

<b>1. FreeRTOS 开发环境搭建</b>	<b>1</b>
1.1 SDK 目录介绍	1
1.2 开发环境搭建	2
1.2.1 Windows 环境搭建	2
1.2.1.1 所需工具	2
1.2.1.2 环境搭建操作步骤	2
1.2.2 Linux 环境搭建	3
1.3 SDK 编译	4
1.3.1 交叉编译工具链配置	4
1.3.2 编译工程	4
<b>2 AliOS 平台开发环境搭建</b>	<b>6</b>
2.1 SDK 目录介绍	6
2.2 开发环境搭建	7
2.2.1 Linux 终端环境搭建	7
2.2.2 Linux Docker 环境搭建	10
2.2.3 Windows 环境搭建	10
2.3 SDK 编译	11
2.3.1 SDK 软件信息	11
2.3.2 ASR BSP 适配信息	11
2.3.3 编译命令	11
<b>3 HarmonyOS 平台开发环境搭建</b>	<b>13</b>
3.2 SDK 目录介绍	13
3.3 开发环境搭建	14
3.3.1 Linux 环境搭建	14
3.3.2 Windows 环境搭建	14
3.4 SDK 编译	14
3.4.1 SDK 软件信息	14
3.4.2 编译命令	15
<b>A. 附录-Docker 环境搭建</b>	<b>18</b>
A.1 Docker 安装	18
A.1.1 Windows 下 Docker 环境搭建	18
A.1.1.1 下载 Docker Desktop	18
A.1.1.2 安装 Docker Desktop	18
A.1.1.3 Docker 镜像加载	22
A.1.2 Linux 下 Docker 环境搭建	23
A.1.2.1 安装 Docker 编译环境	23
A.2 退出 Docker 环境	24
A.3 再次进入 Docker 环境	24

# 表格

表 1-1 FreeRTOS SDK 目录 .....	1
表 2-1 AliOS SDK 目录.....	6
表 3-1 HarmonyOS SDK 目录.....	13

ASR Confidential

# 插图

图 1-1 make 工具 .....	2
图 1-2 make 工具文件目录 .....	2
图 1-3 将 make 文件复制到 Git 对应目录中 .....	3
图 1-4 验证 make 命令 .....	3
图 1-5 安装 make 工具 .....	3
图 1-6 验证 make 命令 .....	3
图 1-7 自动配置工具链及环境 .....	4
图 1-8 清除工程 .....	4
图 1-9 编译工程 .....	5
图 1-10 固件路径 .....	5
图 2-1 安装 python2.7 .....	8
图 2-2 验证 python 是否安装成功 .....	8
图 2-3 安装 pip 工具 .....	8
图 2-4 下载 pip 安装脚本 .....	8
图 2-5 执行脚本安装 pip 工具 .....	9
图 2-6 安装 gcc-multilib .....	9
图 2-7 安装 make .....	9
图 2-8 创建软链接 .....	9
图 2-9 安装 python 依赖包 .....	10
图 2-10 安装 aos-cube 工具链 .....	10
图 2-11 验证 aos-cube 是否安装成功 .....	10
图 2-12 编译 SDK .....	11
图 2-13 编译成功 .....	12
图 3-1 进入 SDK 根目录 .....	15
图 3-2 验证 hb 工具是否正常 .....	15
图 3-3 运行 hb 工具错误 .....	16
图 3-4 卸载 hb 工具 .....	16
图 3-5 重新安装 hb 工具 .....	16
图 3-6 设置 ohos 编译根目录 .....	16
图 3-7 选择对应编译平台 .....	17
图 3-8 编译工程 .....	17
图 3-9 编译成功 .....	17
图 3-10 固件路径 .....	17
图 A-1 下载 Docker Desktop for Windows .....	18
图 A-2 Docker DeskTOP 安装包 .....	18
图 A-3 选择 WSL2 .....	19
图 A-4 安装成功, 重启电脑 .....	19
图 A-5 启动 Docker Desktop .....	20
图 A-6 应用启动成功 .....	20
图 A-7 右键点击开始菜单 .....	21
图 A-8 PowerShell 界面 .....	21

图 A-9 命令执行结果 .....	21
图 A-10 获取 ASR Docker 镜像 .....	22
图 A-11 启动带共享目录 Docker 环境 .....	22
图 A-12 查看共享目录下文件 .....	22
图 A-13 进入 SDK 文件夹 .....	22
图 A-14 安装 Docker .....	23
图 A-15 获取 Openharmony Docker 镜像 .....	23
图 A-16 进入 Docker 构建环境 .....	23
图 A-17 退出当前 Docker 环境 .....	24
图 A-18 再次进入 Docker 环境 .....	24

ASR Confidential

# 1. FreeRTOS 开发环境搭建

## 1.1 SDK 目录介绍

SDK 目录结构如表 1-1 所示：

表 1-1 FreeRTOS SDK 目录

文件夹名	描述
	<p><b>at_cmd</b>: AT 指令用户层的源代码文件夹</p> <p><b>build</b>: 编译环境相关资源及介绍</p> <p><b>cloud</b>: 云端开发相关目录</p> <p><b>common</b>: Lega SDK 使用的通用文件</p> <p><b>demo</b>: ASR550X 的 Wi-Fi、外设和加密算法示例工程</p> <p><b>doc</b>: SDK 相关文档</p> <p><b>freertos</b>: 基于 FreeRTOS V9.0.0 的系统源码</p> <p><b>lib</b>: ASR582X Wi-Fi/BLE 协议栈及外设的库文件和相关头文件</p> <p><b>lwip</b>: 基于 LwIP V2.1.2 定制的 TCP/IP 协议栈源码</p> <p><b>peripheral</b>: 为平台外设相关源文件</p> <p><b>platform</b>: 平台相关文件夹</p> <p><b>tools</b>: 相关工具文件夹</p> <p><b>version</b>: 版本信息</p>



## 1.2 开发环境搭建

### 1.2.1 Windows 环境搭建

#### 1.2.1.1 所需工具

Git Bash

#### 1.2.1.2 环境搭建操作步骤

1. 安装 Git Bash
2. Git Bash 支持 make

默认的 Git Bash 不支持 make 命令，如果需要使用 make，解决步骤如下：

- 到 <https://sourceforge.net/projects/ezwinports/files/> 去下载 make-4.3-without-guile-w32-bin.zip 这个文件

<a href="#">source-highlight-3.1.9-w32-src.zip</a>	2021-09-16	3.9 MB
<a href="#">make-4.3-with-guile-w32-bin.zip</a>	2020-01-25	6.6 MB
<a href="#">make-4.3-without-guile-w32-bin.zip</a>	2020-01-25	361.5 kB
<a href="#">make-4.3-w32-src.zip</a>	2020-01-25	2.6 MB

图 1-1 make 工具

- 将文件解压，解压后文件目录如下：

Download > make-4.3-without-guile-w32-bin >	
名称	修改日期
bin	2020/1/25 20:57
include	2020/1/25 20:56
lib	2020/1/25 20:55
share	2020/1/25 20:53

图 1-2 make 工具文件目录

- 将解压出来的所有文件拷贝到 Git 的安装目录 mingw64 目录内，如果有弹窗提示需要替换文件选择跳过或不替换



图 1-3 将 make 文件复制到 Git 对应目录中

- 此时 Git Bash 就可以使用 make 命令了

```
$ make --version
GNU Make 4.3
Built for Windows32
Copyright (C) 1988-2020 Free Software Foundation, Inc.
```

图 1-4 验证 make 命令

## 1.2.2 Linux 环境搭建

Linux 环境下使用默认终端。

- 安装 make 依赖包

执行指令：`sudo apt-get -y install make`

```
gongjh@gongjh:~$ sudo apt-get -y install make
[sudo] password for gongjh:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

图 1-5 安装 make 工具

- 验证 make 命令是否安装成功

执行指令：`make --version`

```
gongjh@gongjh:~$ make --version
GNU Make 4.2.1
Built for x86_64-pc-linux-gnu
Copyright (C) 1988-2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

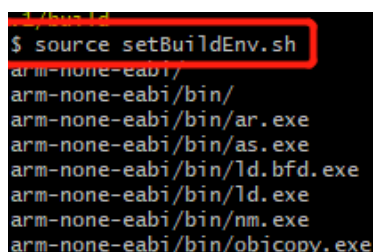
图 1-6 验证 make 命令

## 1.3 SDK 编译

### 1.3.1 交叉编译工具链配置

- 进入 SDK 根目录下的 build 目录。
- 执行 build 目录下的 setBuildEnv.sh 脚本，会自动解压 SDK/tools/toolchain 目录的压缩包到压缩包所在的当前目录，并且自动配置工具链环境变量 TOOLCHAIN\_PATH。

执行指令：`source setBuildEnv.sh`



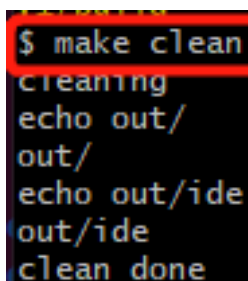
```
$ source setBuildEnv.sh
arm-none-eabi/
arm-none-eabi/bin/
arm-none-eabi/bin/ar.exe
arm-none-eabi/bin/as.exe
arm-none-eabi/bin/ld.bfd.exe
arm-none-eabi/bin/ld.exe
arm-none-eabi/bin/nm.exe
arm-none-eabi/bin/objcopy.exe
```

图 1-7 自动配置工具链及环境

### 1.3.2 编译工程

- 1) 清除编译产生的文件，使用命令：

`make clean`



```
$ make clean
cleaning
echo out/
out/
echo out/ide
out/ide
clean done
```

图 1-8 清除工程

- 2) 编译命令：

- a) 编译 app

- 编译 5822s 的 app，编译命令为：

`make TARGET=duet_demo ic_type=5822s`

如果需要加入 mesh server 功能，编译命令为：

`make TARGET=duet_demo ic_type=5822s version=mesh_generic_onoff_server`

如果需要加入 mesh client 功能，编译命令为：

```
make TARGET=duet_demo ic_type=5822s version=mesh_generic_onoff_client
```

- 编译 5822t 的 app，编译命令为：

```
make TARGET=duet_demo ic_type=5822t
```

如果需要加入 mesh server 功能，编译命令为：

```
make TARGET=duet_demo ic_type=5822t version=mesh_generic_onoff_server
```

如果需要加入 mesh client 功能，编译命令为：

```
make TARGET=duet_demo ic_type=5822t version=mesh_generic_onoff_client
```

- 编译 5822n 的 app，编译命令为：

```
make TARGET=duet_demo ic_type=5822n
```

- 编译 5822c 的 app，编译命令为：

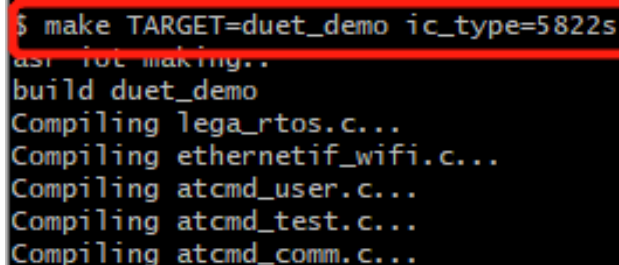
```
make TARGET=duet_demo ic_type=5822c
```

b) 编译外设和加密模块：

```
make TARGET=duet_demo/peripheral/adc
```

```
make TARGET=duet_demo/security/aes
```

其中 TAGRET 选项为所要编译的目标工程，ic\_type 选项为所选芯片型号。



```
$ make TARGET=duet_demo ic_type=5822s
as: not making..
build duet_demo
Compiling lega_rtos.c...
Compiling ethernetif_wifi.c...
Compiling atcmd_user.c...
Compiling atcmd_test.c...
Compiling atcmd_comm.c...
```

图 1-9 编译工程

- 3) 编译完成后，生成的相应的 bin 文件位于 *build/out/duet\_demo* 目录下。

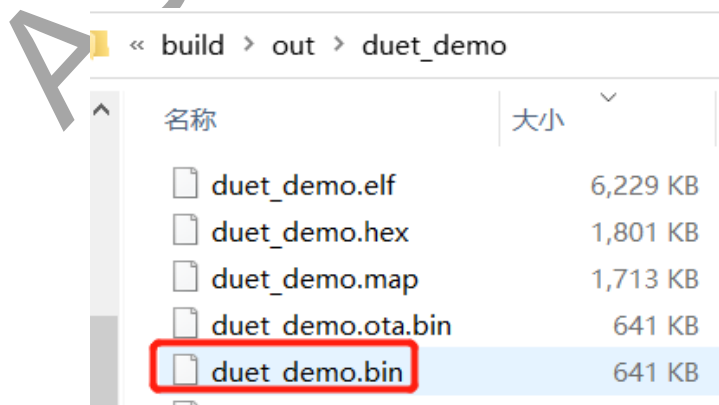


图 1-10 固件路径

## 2

## AliOS 平台开发环境搭建

## 2.1 SDK 目录介绍

表 2-1 AliOS SDK 目录

文件夹名	描述
<ul style="list-style-type: none"> <li>✓ ASR582X_ALIOS-THINGS               <ul style="list-style-type: none"> <li>&gt; doc_tool</li> <li>✓ Living_SDK                   <ul style="list-style-type: none"> <li>&gt; 3rdparty</li> <li>&gt; app</li> <li>&gt; board</li> <li>&gt; build</li> <li>&gt; device</li> <li>&gt; doc</li> <li>&gt; example</li> <li>&gt; framework</li> <li>&gt; include</li> <li>&gt; kernel</li> <li>&gt; platform</li> <li>&gt; projects</li> <li>&gt; security</li> <li>&gt; site_scons</li> <li>&gt; test</li> <li>&gt; tools</li> <li>&gt; utility</li> <li>🔑 LICENSE</li> <li>☰ NOTICE</li> <li>📄 README.md</li> <li>🔗 ucube.py</li> <li>&gt; Products</li> <li>&gt; Service</li> <li>&gt; tools</li> <li>💰 build.sh</li> <li>🔑 LICENSE</li> <li>📄 README.md</li> </ul> </li> </ul> </li> </ul>	<p><b>doc_tool</b>: ASR 提供的编译文档和烧录工具</p> <p><b>Living_SDK</b>: AliOS v1.3.4 SDK</p> <ul style="list-style-type: none"> <li><b>__3rdparty</b>: 第三方开源组件</li> <li><b>__app</b>: 系统调用类 API 组件</li> <li><b>__board</b>: 放置板级相关启动、配置、初始化代码，以及板级外设驱动</li> <li><b>__build</b>: 编译构建相关工具和脚本</li> <li><b>__device</b>: 外设驱动文件</li> <li><b>__doc</b>: AliOS 相关文档</li> <li><b>__example</b>: 示例代码</li> <li><b>__framework</b>: IoT 通用组件</li> <li><b>__include</b>: 组件对外头文件</li> <li><b>__kernel</b>: 内核及相关组件，包括 Rhino,协议栈等</li> <li><b>__platform</b>: 芯片 CPU 架构相关的调度代码和 BSP</li> <li><b>__projects</b>: 为不同开发环境提供的工程相关文件</li> <li><b>__security</b>: 安全类组件</li> <li><b>__site_scons</b>: scons 自动化构建工具文件夹</li> <li><b>__test</b>: 单元测试文件夹</li> <li><b>__tools</b>: cli、at 等控制台类工具</li> <li><b>__utility</b>: IoT 通用库，例如 cJSON</li> </ul> <p><b>Products</b>: 阿里生活物联网示例工程</p> <p><b>Service</b>: 云端开发相关目录</p> <p><b>tools</b>: 相关芯片执行脚本</p>

## 2.2 开发环境搭建

ASR582X\_AliOS-Things SDK 支持 Linux 终端环境、Linux Docker 环境和 Windows Docker 环境下编译

- Linux 终端环境使用系统自带的 Terminal 终端来进行命令行编译，需要按步骤安装环境依赖包。
- Linux Docker 环境和 Windows Docker 环境使用 docker 容器，运行 ASR 提供的通用镜像来进行编译，镜像已适配 ASR 各版 SDK 编译环境，简化了用户编译过程，需要安装 docker。

### 2.2.1 Linux 终端环境搭建

1. 推荐您安装 64 位 Ubuntu 系统。
2. 安装 Ubuntu 依赖软件包。

```
#安装 python 和 pip, Alios 仅支持 python2.7
sudo apt-get install -y python2

#如果您使用的 Ubuntu 版本为 20.04，则无法使用如下命令安装 pip
sudo apt-get install -y python-pip

#如果您的 Ubuntu 版本为 20.04，安装 pip 工具使用如下指令
curl https://bootstrap.pypa.io/pip/2.7/get-pip.py --output get-pip.py
sudo python2 get-pip.py

#安装 gcc 依赖包
sudo apt-get -y install gcc-multilib

#安装 make 工具
sudo apt-get -y install make

#创建 python2 软链接为 python
sudo ln -s /usr/bin/python2 /usr/bin/python

#安装依赖库和 aos-cube
sudo python -m pip install wheel
sudo python -m pip install aos-cube
```

#### a) 安装 python2.7

执行指令：`sudo apt-get install -y python2`

```
jinghong@jinghong-virtual-machine:~/Documents/ASRCode$ sudo apt-get install -y python2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libpython2.7-minimal libpython2.7-stdlib python2-minimal python2.7 python2.7-minimal
Suggested packages:
  python2-doc python-tk python2.7-doc binutils binfmt-support
The following NEW packages will be installed:
  libpython2.7-minimal libpython2.7-stdlib python2 python2-minimal python2.7 python2.7-minimal
0 upgraded, 7 newly installed, 0 to remove and 70 not upgraded.
Need to get 4,005 kB of archives.
After this operation, 16.2 MB of additional disk space will be used.
```

**安装python2.7**

图 2-1 安装 python2.7

b) 验证 python 是否安装成功

执行指令：`python2 --version`，必须为 2.7 版本

```
gongjh@gongjh-virtual-machine:~/Desktop$ python2 --version
Python 2.7.18
gongjh@gongjh-virtual-machine:~/Desktop$
```

图 2-2 验证 python 是否安装成功

c) 安装 pip 工具

执行指令：`sudo apt-get install -y python-pip`

```
gongjh@gongjh-virtual-machine:~/Desktop$ sudo apt-get install -y python-pip
[sudo] password for gongjh:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential dpkg-dev
  fakeroot g++ g++-11 gcc gcc-11 gcc-12-base libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan6 libatomic1
  libbinutils libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev
  libctf-nobfd0 libctf0 libdpkg-perl libfakeroot libfile-fcntllock-perl
  libgcc-11-dev libgcc-s1 libgomp1 libitm1 liblsan0 libnsl-dev
```

图 2-3 安装 pip 工具

如您使用的 Ubuntu 版本为 20.04，安装 pip 失败的话，您可使用如下指令来安装 pip 工具。

执行指令：`curl https://bootstrap.pypa.io/pip/2.7/get-pip.py --output get-pip.py`

```
gongjh@gongjh:~$ curl https://bootstrap.pypa.io/pip/2.7/get-pip.py --output get-pip.py
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 1863k 100 1863k 0 0 984k 0 0:00:01 0:00:01 --:--:-- 984k
```

图 2-4 下载 pip 安装脚本

执行指令：`sudo python2 get-pip.py`



```
gongjh@gongjh:~$ sudo python2 get-pip.py
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade
details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/dev
Collecting pip<21.0
  Downloading pip-20.3.4-py2.py3-none-any.whl (1.5 MB)
    | 1.5 MB 298 kB/s
Collecting setuptools<45
  Downloading setuptools-44.1.1-py2.py3-none-any.whl (583 kB)
    | 583 kB 2.3 MB/s
Collecting wheel
  Downloading wheel-0.37.1-py2.py3-none-any.whl (35 kB)
Installing collected packages: pip, setuptools, wheel
Successfully installed pip-20.3.4 setuptools-44.1.1 wheel-0.37.1
```

图 2-5 执行脚本安装 pip 工具

## d) 安装 gcc 依赖包

执行指令: `sudo apt-get -y install gcc-multilib`

```
gongjh@gongjh:~/Documents/ASRCode/ASR582X_AliOS-Things$ sudo apt-get -y install gcc-multilib
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  gcc-11-multilib lib32asan6 lib32atomic1 lib32gcc-11-dev lib32gcc-s1 lib32gomp1 lib32lttng lib32quadmath0 lib32stdc++6 lib32ubsan1
  lib32zstd1
The following NEW packages will be installed:
  gcc-11-multilib gcc-multilib lib32asan6 lib32atomic1 lib32gcc-11-dev lib32gcc-s1 lib32gomp1 lib32lttng lib32quadmath0 lib32stdc++6 lib32ubsan1
0 upgraded, 11 newly installed, 0 to remove and 66 not upgraded.
Need to get 21.8 MB of archives.
After this operation, 82.7 MB of additional disk space will be used.
Get:1 http://cn.archive.ubuntu.com/ubuntu jammy-updates/main amd64 lib32asan6 amd64 2.35-0ubuntu3.1 [2,837 kB]
Get:2 http://cn.archive.ubuntu.com/ubuntu jammy-updates/main amd64 lib32atomic1 amd64 2.35-0ubuntu3.1 [1,444 kB]
Get:3 http://cn.archive.ubuntu.com/ubuntu jammy-updates/main amd64 lib32gcc-s1 amd64 2.35-0ubuntu3.1 [2,978 kB]
Get:4 http://cn.archive.ubuntu.com/ubuntu jammy-updates/main amd64 lib32gomp1 amd64 2.35-0ubuntu3.1 [1,632 kB]
Get:5 http://cn.archive.ubuntu.com/ubuntu jammy-updates/main amd64 lib32quadmath0 amd64 12.1.0-2ubuntu1-22.04 [64.5 kB]
Get:6 http://cn.archive.ubuntu.com/ubuntu jammy-updates/main amd64 lib32stdc++6 amd64 12.1.0-2ubuntu1-22.04 [54.4 kB]
Get:7 http://cn.archive.ubuntu.com/ubuntu jammy-updates/main amd64 lib32ubsan1 amd64 12.1.0-2ubuntu1-22.04 [133 kB]
Get:8 http://cn.archive.ubuntu.com/ubuntu jammy-updates/main amd64 gcc-11-multilib amd64 12.1.0-2ubuntu1-22.04 [127 kB]
Get:9 http://cn.archive.ubuntu.com/ubuntu jammy-updates/main amd64 gcc-multilib amd64 12.1.0-2ubuntu1-22.04 [132.0 kB]
debconf: delaying package configuration, since apt-utils is not installed
Fetched 21.8 MB in 1s (20.4 MB/s)
Selecting previously unselected package gcc-11-multilib.
(Reading database ... 123456789 files and directories currently installed.)
Preparing to unpack .../gcc-11-multilib_12.1.0-2ubuntu1-22.04_amd64.deb ...
Unpacking gcc-11-multilib (12.1.0-2ubuntu1-22.04) ...
Selecting previously unselected package gcc-multilib.
Preparing to unpack .../gcc-multilib_12.1.0-2ubuntu1-22.04_amd64.deb ...
Unpacking gcc-multilib (12.1.0-2ubuntu1-22.04) ...
Setting up gcc-11-multilib (12.1.0-2ubuntu1-22.04) ...
Setting up gcc-multilib (12.1.0-2ubuntu1-22.04) ...
```

图 2-6 安装 gcc-multilib

## e) 安装 make 依赖包

执行指令: `sudo apt-get -y install make`

```
gongjh@gongjh:~/Documents/ASRCode/ASR582X_AliOS-Things$ sudo apt-get -y install make
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  make-doc
The following NEW packages will be installed:
  make
0 upgraded, 1 newly installed, 0 to remove and 42 not upgraded.
Need to get 162 kB of archives.
After this operation, 393 kB of additional disk space will be used.
Get:1 http://cn.archive.ubuntu.com/ubuntu jammy-updates/main amd64 make amd64 4.2.1-2ubuntu1 [162 kB]
debconf: delaying package configuration, since apt-utils is not installed
Fetched 162 kB in 1s (16.2 MB/s)
Selecting previously unselected package make.
(Reading database ... 123456789 files and directories currently installed.)
Preparing to unpack .../make_4.2.1-2ubuntu1_amd64.deb ...
Unpacking make (4.2.1-2ubuntu1) ...
Setting up make (4.2.1-2ubuntu1) ...
```

图 2-7 安装 make

## f) 创建软链接

安装 python2 后可执行文件名为 python2, 而 SDK 使用的可执行文件名为 python, 所以需要将 python2 软链接为 python 以供 SDK 使用。

执行指令: `sudo ln -s /usr/bin/python2 /usr/bin/python`

```
sudo: python: command not found
jinghong@jinghong-virtual-machine:~/Documents/ASRCode$ sudo ln -s /usr/bin/python2 /usr/bin/python
```

图 2-8 创建软链接



## g) 安装 python 依赖包

执行指令: `sudo python -m pip install wheel`

```
jinghong@jinghong-virtual-machine:~/Documents/ASRCode$ sudo python -m pip install wheel
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained.
details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support
Collecting wheel
  Downloading wheel-0.37.1-py2.py3-none-any.whl (35 kB)
Installing collected packages: wheel
Successfully installed wheel-0.37.1
```

安装python依赖包-wheel

图 2-9 安装 python 依赖包

## h) 安装 aos-cube 工具链

执行指令: `sudo python -m pip install aos-cube`

```
jinghong@jinghong-virtual-machine:~/Documents/ASRCode$ sudo python -m pip install aos-cube
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained.
details about Python 2 support in pip can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support
Collecting aos-cube
  Downloading aos-cube-0.5.11.tar.gz (52 kB)
    | 52 kB 249 kB/s
Collecting click
  Downloading click-7.1.2-py2.py3-none-any.whl (82 kB)
    | 82 kB 510 kB/s
Collecting configparser
  Downloading configparser-4.0.2-py2.py3-none-any.whl (22 kB)
Collecting esptool
  Downloading esptool-3.3.1.tar.gz (213 kB)
    | 213 kB 1.9 MB/s
Collecting paho-mqtt
```

安装aos-cube工具链

图 2-10 安装 aos-cube 工具链

## i) 验证 aos-cube 是否安装成功

执行指令: `aos --version`

```
jinghong@jinghong-virtual-machine:~/Documents/ASRCode/ASR582X_AliOS-Things$ aos --version
0.5.11
```

查看aos-cube是否安装成功

图 2-11 验证 aos-cube 是否安装成功

## 2.2.2 Linux Docker 环境搭建

详细步骤见附录 A1.2 Linux 下 Docker 环境搭建。

## 2.2.3 Windows 环境搭建

详细步骤见附录 A1.1 Windows 下 Docker 环境搭建。

## 2.3 SDK 编译

### 2.3.1 SDK 软件信息

- 基于阿里生活物联网平台 SDK V1.5.01 (AliOS-1.3.4)。
- 搭配 GCC Version 5.4.1 toolchain (gcc-arm-none-eabi-5\_4-2016q3-20160926), SDK 已经集成此版 toolchain, 在 *Living\_SDK\build\compiler\gcc-arm-none-eabi\Linux64* 下。
- 默认工程是 AliOS comboapp, 此工程支持 BLE 辅助配网/一键配网。
- 提供外设驱动模块并适配 AliOS HAL。

### 2.3.2 ASR BSP 适配信息

*Living\_SDK\board\asr5821*: board 适配

*Living\_SDK\platform\mcu\asr5821*: 包含外设驱动 driver/hal 适配, 芯片 platform 适配以及 WIFI/BLE combo library 等

### 2.3.3 编译命令

支持编译不同芯片类型固件, 在 SDK 根目录下执行脚本命令, 参考指令如下:

<code>./build.sh ic_type=5822S</code>	---build 5822S image
<code>./build.sh ic_type=5822N</code>	---build 5822N image
<code>./build.sh ic_type=5822C</code>	---build 5822C image
<code>./build.sh clean</code>	---clean compile

暂不支持 5822T 编译

生成的镜像文件为 `out\comboapp@asr5821\comboapp@asr5821.bin`



```
jinghong@jinghong-virtual-machine:~/Documents/ASRCode/ASR582X_AliOS-Things$ ./build.sh ic_type=5822S
REGION SET AS MAINLAND
ENV SET AS ONLINE
CONFIG_DEBUG SET AS 0
ARGS SET AS NULL
-----
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/home/jinghong/Documents/ASRCode/ASR582X_A
one-eabi/Linux64/bin/
OS: Linux
Product type=example app_name=comboapp board=asr5821 REGION=MAINLAND ENV=ONLINE CONFIG_DEBUG=0 ARGS=ic_type=5822S
-----
do cp
/home/jinghong/Documents/ASRCode/ASR582X_AliOS-Things
make aos sdk
aos-cube version: 0.5.11
Cleaning...
Done
aos-cube version: 0.5.11
Making config file for first time
processing components: comboapp asr5821 platform/mcu/asr5821 vcall init auto_component
server region: MAINLAND
server env: ONLINE
APP: comboapp Board: asr5821
```

图 2-12 编译 SDK

```

| comboapp | 11955 | 1033 |
| digest_algorithm | 24 | 0 |
| framework | 344 | 12 |
| hal | 370 | 12 |
| imbedtts | 26688 | 88 |
| iotx-hal | 8788 | 228 |
| kernel_init | 1159 | 48 |
| libaiotss | 682 | 0 |
| libasr_combo | 416656 | 105075 |
| libc | 51604 | 1546 |
| libgcc | 3608 | 0 |
| libiot_sdk | 132914 | 3321 |
| libm | 4753 | 1 |
| libota_port | 2070 | 326 |
| log | 567 | 20 |
| net | 60601 | 2577 |
| netmgr | 2236 | 246 |
| newlib_stub | 770 | 0 |
| ota | 1737 | 1 |
| ota_download | 1531 | 12 |
| ota_hal | 997 | 4 |
| ota_transport | 1247 | 24 |
| ota_verify | 2229 | 12 |
| rhino | 16099 | 4685 |
| vcall | 1470 | 4 |
| vfs | 1881 | 1209 |
| yloop | 1201 | 24 |
| *fill* | 1024 | 298 |
|=====|
| TOTAL (bytes) | 832212 | 126016 |
|=====|
image size:0xcea94
image crc:0xaf65
compress image size:0x78570
compress image crc:0xf3c3
846484
9eea9f957d0535e84b15c7924df1d0b2
build time is 0min 48s
firmware_comboapp@asr5821.bin size=827K
jtinghong@jtinghong-virtual-machine: /documents/ASRCode/ASR582X_AliOS-Things$ c
```

图 2-13 编译成功

## 3 HarmonyOS 平台开发环境搭建

### 3.2 SDK 目录介绍

表 3-1 HarmonyOS SDK 目录

文件夹名	描述
▼ HARMONYLO_ASR	
> applications	<b>applications:</b> 应用程序示例
> base	<b>base:</b> 基础软件服务子系统集&硬件服务子系统集
> build	<b>build:</b> 组件化编译、构建和配置脚本
> developeptools	<b>developeptools:</b> 云端开发相关目录
> device	<b>device:</b> 相关芯片执行脚本
> docs	<b>docs:</b> 介绍、说明文档
> domains	<b>domains:</b> 增强软件服务子系统集
> drivers	<b>drivers:</b> 驱动子系统
> foundation	<b>foundation:</b> 系统基础能力子系统集
> kernel	<b>kernel:</b> 内核子系统, 包含 Liteos-a()/Liteos-m
> prebuilts	<b>prebuilts:</b> 编译器及工具链子系统
> test	<b>test:</b> 测试子系统
> third_party	<b>third_party:</b> 第三方开源组件
> utils	<b>utils:</b> 常用的工具集, 如 KV 存储、文件操作、定时器等
> vendor	<b>vendor:</b> 厂商提供的软件
{ } ohos_config.json	<b>ohos_config.json:</b> 自动化构建配置文件

## 3.3 开发环境搭建

### 3.3.1 Linux 环境搭建

详细步骤见[附录 A.1.2 Linux 下 Docker 环境搭建](#)。

### 3.3.2 Windows 环境搭建

详细步骤见[附录 A.1.1 Windows 下 Docker 环境搭建](#)。

## 3.4 SDK 编译

### 3.4.1 SDK 软件信息

- ASR 提供两版 SDK：
  - 1) harmonyl0\_asr 适配鸿蒙 version1.1.3 LTL
  - 2) openharmony3.0\_asr 适配 OpenHarmony v3.0 LTS
- harmony SDK 编译采用构建命令行工具 hb 来完成，其本质是一个 python3 实现的脚本，需要 Python3.7.4 及以上版本支持。其作用是：
  - 1) hb set: 设置 OpenHarmony 源码目录和要编译的产品
  - 2) hb build: 编译产品、开发板或者组件。解决方案编译实现如下：
    - 读取开发板配置：主要包括开发板使用的编译工具链、编译链接命令和选项等。
    - 调用 gn: 调用 gn gen 命令，读取产品配置（主要包括开发板、内核、选择的组件等）生成解决方案 out 目录和 ninja 文件。
    - 调用 ninja: 调用 ninja -C out/company/product 启动编译。
    - 系统镜像打包：将组件编译产物打包，制作文件系统镜像。

### 3.4.2 编译命令

```
#查看 hb 工具
hb -h

#设置当前目录为 root 根目录
hb set -root .

#设置编译平台
hb set

#编译固件
hb build -f
```

a) 进入 SDK 源码目录

```
root@80fb178497cc:/home/asriot# ls
ASR582X_AliOS-Things  harmonyl0_asr  openharmony3.0_asr
root@80fb178497cc:/home/asriot# cd harmonyl0_asr/
root@80fb178497cc:/home/asriot/harmony0_asr# hb -h
```

图 3-1 进入 SDK 根目录

b) 查看 Docker 环境内 hb 工具是否正常运行

hb 是鸿蒙编译构建工具（hb 是 ohos-build 的简称，而 ohos 又是 openharmony os 的简称）

在 Docker 环境下运行指令：hb -h

```
root@643dd864003f:/home/openharmony# hb -h
usage: hb

OHOS build system

positional arguments:
  {build,set,env,clean,deps}
  build                Build source code
  set                  OHOS build settings
  env                  Show OHOS build env
  clean                Clean output
  deps                 OHOS components deps

optional arguments:
  -h, --help            show this help message and exit
```

图 3-2 验证 hb 工具是否正常

如果 hb 指令运行出错，则需要当前 Docker 环境下重新安装 hb 工具，在 Docker 环境下执行：

```
#首先需要 cd 到源码根目录（OpenHarmony 的 Docker 镜像内已包含 pip 工具）
pip3 uninstall ohos-build
pip3 install build/lite
```

```

root@d4634485cafb:/home/openharmony# hb -h
Traceback (most recent call last):
  File "/usr/local/bin/hb", line 11, in <module>
    sys.exit(main())
  File "/usr/local/lib/python3.8/dist-packages/hb/__main__.py", line 49, in main
    topdir = find_top()
  File "/usr/local/lib/python3.8/dist-packages/hb/__main__.py", line 37, in find_top
    raise Exception("Please call hb utilities inside source root directory")
Exception: Please call hb utilities inside source root directory

```

图 3-3 运行 hb 工具错误

```

root@d4634485cafb:/home/openharmony# pip3 uninstall ohos_build
Uninstalling ohos-build-0.4.6:
  /usr/local/bin/hb
  /usr/local/lib/python3.8/dist-packages/hb/__init__.py
  /usr/local/lib/python3.8/dist-packages/hb/__main__.py
  /usr/local/lib/python3.8/dist-packages/hb/__pycache__/__init__.cpython-38.pyc
  /usr/local/lib/python3.8/dist-packages/hb/__pycache__/__main__.cpython-38.pyc
  /usr/local/lib/python3.8/dist-packages/ohos_build-0.4.6.dist-info/INSTALLER
  /usr/local/lib/python3.8/dist-packages/ohos_build-0.4.6.dist-info/LICENSE
  /usr/local/lib/python3.8/dist-packages/ohos_build-0.4.6.dist-info/METADATA
  /usr/local/lib/python3.8/dist-packages/ohos_build-0.4.6.dist-info/RECORD
  /usr/local/lib/python3.8/dist-packages/ohos_build-0.4.6.dist-info/WHEEL
  /usr/local/lib/python3.8/dist-packages/ohos_build-0.4.6.dist-info/entry_points.txt
  /usr/local/lib/python3.8/dist-packages/ohos_build-0.4.6.dist-info/top_level.txt
Proceed (y/n)? y
Successfully uninstalled ohos-build-0.4.6

```

图 3-4 卸载 hb 工具

```

root@d4634485cafb:/home/openharmony# pip3 install build/lite/
Processing ./build/lite
Requirement already satisfied: prompt_toolkit==1.0.14 in /usr/local/lib/python3.8/dist-packages (from prompt_toolkit==1.0.14)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.8/dist-packages (from prompt_toolkit==1.0.14)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.8/dist-packages (from prompt_toolkit==1.0.14)
Installing collected packages: ohos-build
  Running setup.py install for ohos-build ... done
Successfully installed ohos-build-0.1.1

```

图 3-5 重新安装 hb 工具

- c) 设置当前目录为 harmony 编译根目录

首先需要确认当前目录为 SDK 根目录，然后使用如下命令将当前目录设置为 ohos 的编译根目录：`hb set -root .`

```

root@80fb178497cc:/home/asriot/harmonylo_asr# hb set -root .
root@80fb178497cc:/home/asriot/harmonylo_asr#

```

图 3-6 设置 ohos 编译根目录

- d) 选择需要编译的平台

在 Docker 环境下执行命令 `hb set`，选择编译平台。

出现如图 3-7 的 log 打印后，按上下方向键选择对应的编译平台并回车确认。

```
root@80fb178497cc:/home/asriot/harmonylo_asr# hb set
[OHOS INFO] hb root path: /home/asriot/harmonylo_asr
OHOS Which product do you need? (Use arrow keys)

hisilicon
  wifiot_hispark_pegasus
  ipcamera_hispark_taurus
  ipcamera_hispark_aries

asr
  > asr582x
```

图 3-7 选择对应编译平台

## e) 编译工程

在 Docker 环境下执行命令：`hb build -f`，开始编译。

```
root@80fb178497cc:/home/asriot/harmonylo_asr# hb build -f
[OHOS INFO] WARNING at the command-line "--args":1:408: Build argument has no effect.
[OHOS INFO] ohos_build_compiler_specified="gcc" product_path="/home/asriot/harmonylo_asr/vendor/asr/asr582x" device_path="
n" enable_ohos_kernel_liteos_m_fs = false enable_ohos_kernel_liteos_m_cppsupport = false disable_huks_binary=true disable_
ld_datetime="2022-10-09 09:20:52" ohos_full_compile=true
[OHOS INFO]
^-----
[OHOS INFO] The variable "ohos_build_datetime" was set as a build argument
[OHOS INFO] but never appeared in a declare_args() block in any buildfile.
[Terminal 0]
[OHOS INFO] To view all possible args, run "gn args --list <out_dir>"
[OHOS INFO] The build continued as if that argument was unspecified.
[OHOS INFO] Done. Made 90 targets from 69 files in 1986ms
[OHOS INFO] [1/571] gcc cross compiler obj/base/hiviewdfx/hilog_lite/frameworks/mini/libhilog_lite.hiview_log.o
[OHOS INFO] [2/571] gcc cross compiler obj/base/hiviewdfx/hiview_lite/libhiview_lite.hiview_cache.o
[OHOS INFO] [3/571] gcc cross compiler obj/base/hiviewdfx/hilog_lite/command/libhilog_lite_command.hilog_lite_command.o
[OHOS INFO] [4/571] AR libs/libhilog_lite_command.a
[OHOS INFO] [5/571] gcc cross compiler obj/base/hiviewdfx/hilog_lite/frameworks/mini/libhilog_lite.hiview_log_limit.o
[OHOS INFO] [6/571] gcc cross compiler obj/base/hiviewdfx/hiview_lite/libhiview_lite.hiview_config.o
```

图 3-8 编译工程

```
[OHOS INFO] [562/571] STAMP obj/base/startup/syspara_lite/frameworks/parameter/parameter.stamp
[OHOS INFO] [563/571] STAMP obj/build/lite/ohos.stamp
[OHOS INFO] [564/571] STAMP obj/base/startup/syspara_lite/frameworks/parameter/parameter_notes.stamp
[OHOS INFO] [565/571] gcc cross compiler obj/utils/native/lite/kal/timer/src/kal_timer_static.kal.o
[OHOS INFO] [566/571] STAMP obj/utils/native/lite/kal/timer/kal_timer_static.stamp
[OHOS INFO] [567/571] ACTION //build/lite:gen_rootfs(//build/lite/toolchain:arm-none-eabi)
[OHOS INFO] [568/571] STAMP obj/build/lite/gen_rootfs.stamp
[OHOS INFO] [569/571] LINK ./bin/duetiot_582x.elf
[OHOS INFO] [570/571] ACTION //device/asr/asr582x:image(//build/lite/toolchain:arm-none-eabi)
[OHOS INFO] [571/571] STAMP obj/device/asr/asr582x/image.stamp
[OHOS INFO] /home/asriot/harmonylo_asr/vendor/asr/asr582x/fs.yml not found, stop packing fs. If the product does not
[OHOS INFO] asr582x build success
[OHOS INFO] cost time: 0:01:07
```

图 3-9 编译成功

## f) 固件路径

编译成功后，在根目录会自动生成 out 文件夹，生成的固件在 `out/asr582x/asr582x/bin` 目录下：

```
root@80fb178497cc:/home/asriot/harmonylo_asr# ls out/asr582x/asr582x/bin/
duetiot_582x.bin  duetiot_582x.elf  duetiot_582x.lst  duetiot_582x.map  duetiot_582x.ota.bin  gcc.ld
root@80fb178497cc:/home/asriot/harmonylo_asr# ls out/asr582x/asr582x/bin/
```

图 3-10 固件路径



# A. 附录-Docker 环境搭建

## A.1 Docker 安装

### A.1.1 Windows 下 Docker 环境搭建

#### A.1.1.1 下载 Docker Desktop

Docker Desktop 官方连接: <https://docs.docker.com/desktop/install/windows-install>

注意: 安装 Docker Desktop 需满足:

- Windows 11 64 位: 家庭版或专业版版本号 21H2 或更高版本, 或者企业版或教育版 21H2 或更高版本
- Windows 10 64 位: 家庭版或专业版 21H2 (build 19043) 或更高版本, 或者企业版或教育版 20H2 (build 19042) 或更高版本
- 硬件要求:
  - 64 位带二级地址转换的处理器
  - 4G 系统内存



图 A-1 下载 Docker Desktop for Windows

#### A.1.1.2 安装 Docker Desktop

- a) 双击下载好的安装包



图 A-2 Docker Desktop 安装包

## b) 勾选使用 WSL2 替代 hyper-v

因为 hyper-v 后端仅支持 Windows 专业版，所以为了统一操作，都选择 WSL2 后端来替代。

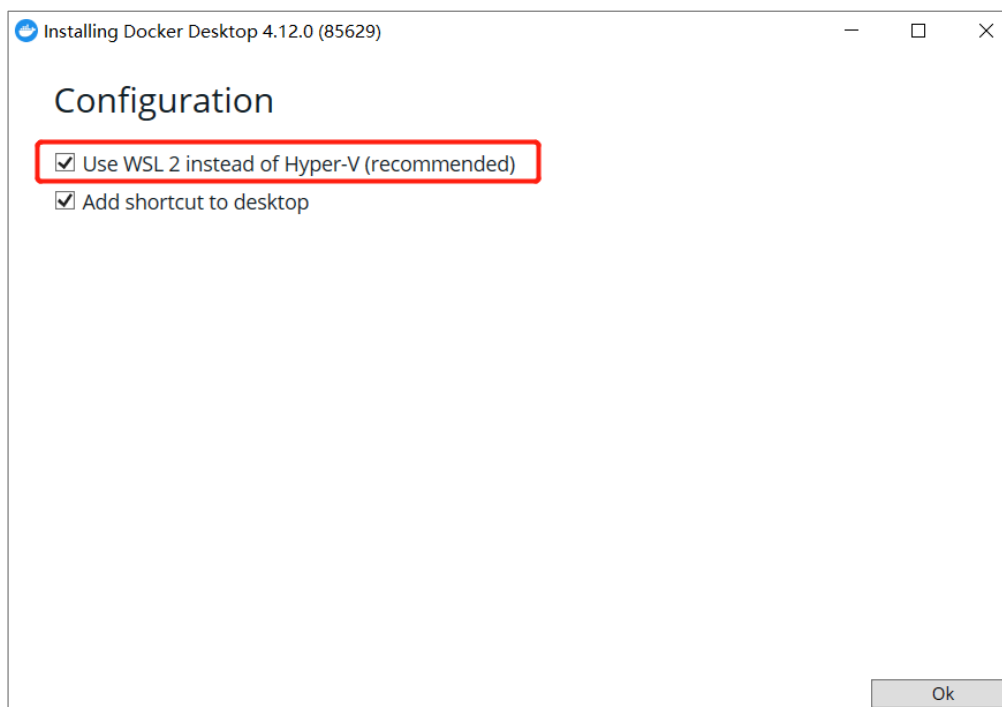


图 A-3 选择 WSL2

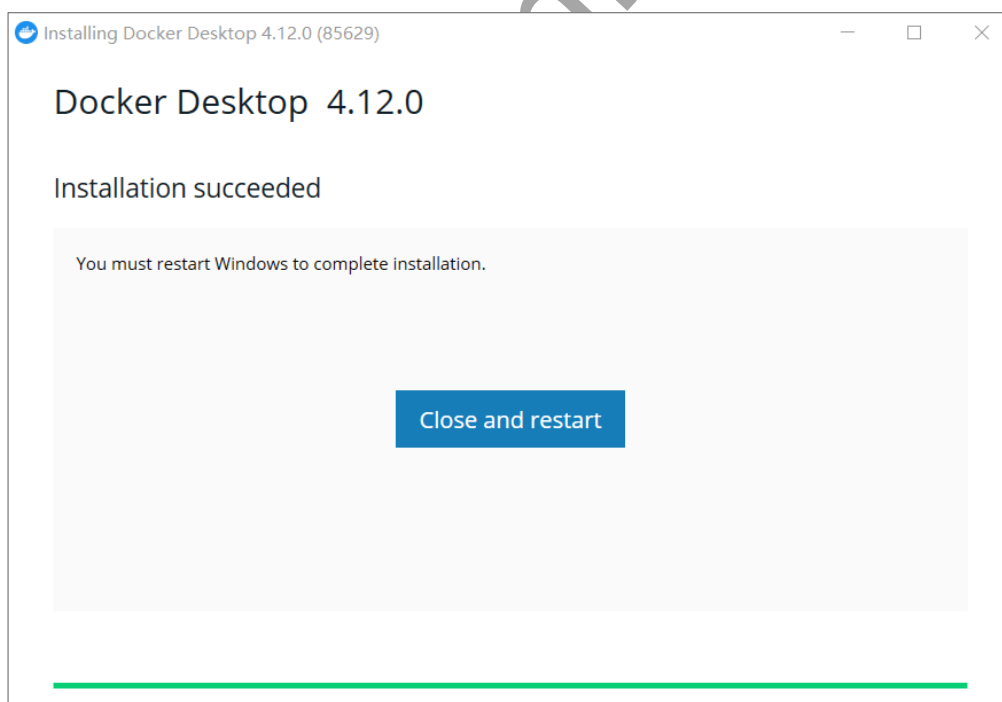


图 A-4 安装成功，重启电脑

## c) 使用管理员模式打开 Docker Desktop

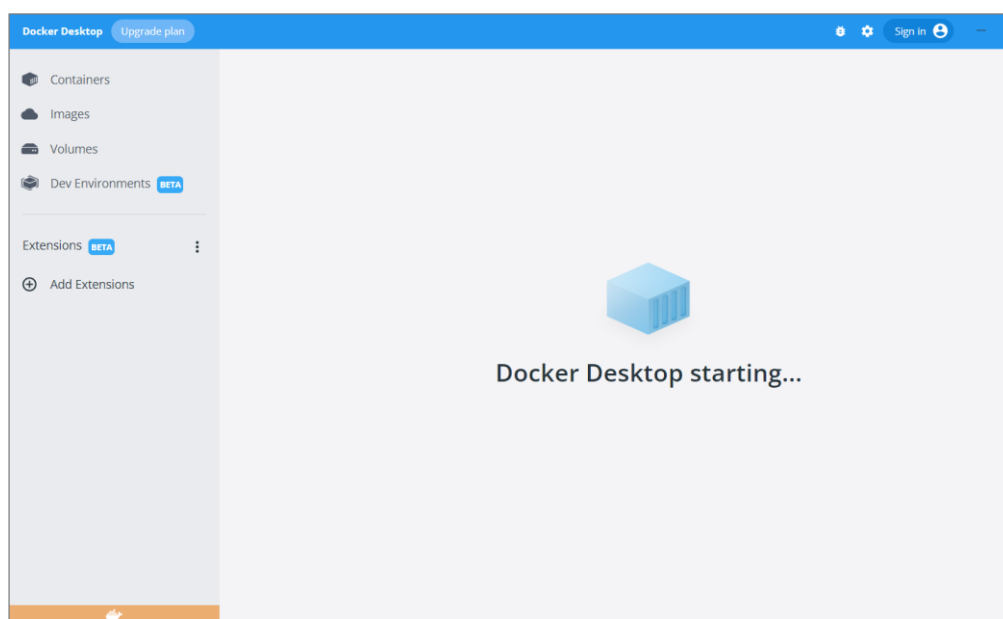


图 A-5 启动 Docker Desktop

## d) Docker Desktop 启动成功

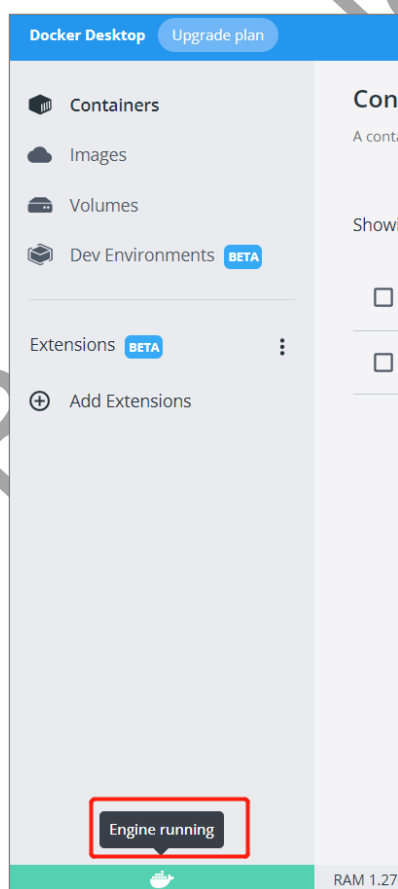


图 A-6 应用启动成功

## e) 验证 Docker 命令

- 以管理员模式启动 Windows PowerShell



图 A-7 右键点击开始菜单



图 A-8 PowerShell 界面

- 运行如下指令，验证 Docker 命令：

在管理员模式下的 PowerShell 下运行：`docker run hello-world`

```
PS C:\Windows\system32> docker run hello-world

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/

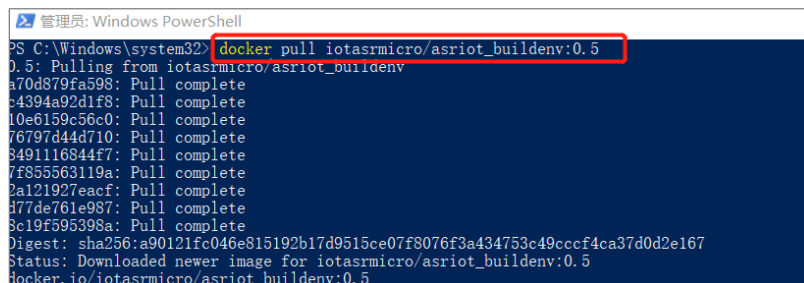
For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

图 A-9 命令执行结果

### A.1.1.3 Docker 镜像加载

#### a) 获取 Docker 镜像

在管理员模式 PowerShell 下执行：`docker pull iotasrmicro/asriot_buildenv:0.5`



```
PS C:\Windows\system32> docker pull iotasrmicro/asriot_buildenv:0.5
0.5: Pulling from iotasrmicro/asriot_buildenv
a70d879fa598: Pull complete
c4394a92d1f8: Pull complete
10e6159c56c0: Pull complete
76797d44d710: Pull complete
8491116844f7: Pull complete
7f855563119a: Pull complete
2a121927eacf: Pull complete
477de761e987: Pull complete
8c19f595398a: Pull complete
Digest: sha256:a90121fc046e815192b17d9515ce07f8076f3a434753c49cccf4ca37d0d2e167
Status: Downloaded newer image for iotasrmicro/asriot_buildenv:0.5
docker.io/iotasrmicro/asriot_buildenv:0.5
```

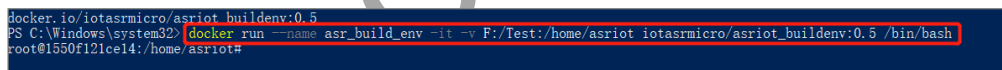
图 A-10 获取 ASR Docker 镜像

#### b) 创建 Docker 容器运行环境

- Docker 环境本质上是一个运行在 Linux 内核的容器，所以如果在 Docker 环境中需要访问 Windows 下的文件，则必须在启动 Docker 时将 Windows 中的指定目录映射到 Docker 环境中相关目录。
- 我们将 SDK 存放的目录指定为共享文件夹，例：SDK 存放目录为 `F:\Test`，在 Docker 启动时，将该目录映射到 Docker 环境中的 `/home/asriot` 目录，执行如下指令启动带共享目录名称为 `asr_build_env` 的 Docker 环境：

```
docker run --name asr_build_env -it -v F:/Test:/home/asriot iotasrmicro/asriot_buildenv:0.5 /bin/bash
```

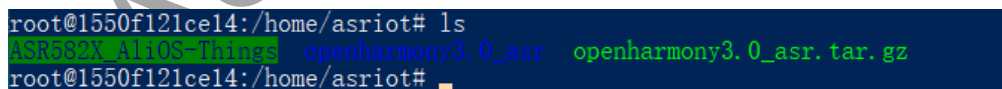
如上指令标红部分为 Windows 下的共享文件夹路径，路径中的斜杠为 `/`，与 Windows 路径中的斜杠相反。



```
docker.io/iotasrmicro/asriot_buildenv:0.5
PS C:\Windows\system32> docker run --name asr_build_env -it -v F:/Test:/home/asriot iotasrmicro/asriot_buildenv:0.5 /bin/bash
root@1550f121ce14:/home/asriot#
```

图 A-11 启动带共享目录 Docker 环境

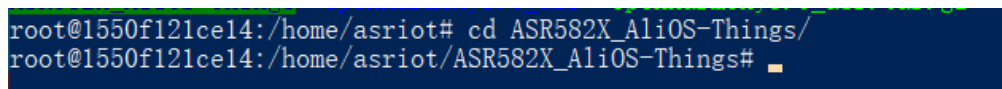
- 此时可以使用 `ls` 命令查看共享目录下的文件



```
root@1550f121ce14:/home/asriot# ls
ASR582X_AliOS-Things  openharmony3.0_asr  openharmony3.0_asr.tar.gz
root@1550f121ce14:/home/asriot#
```

图 A-12 查看共享目录下文件

- 使用 `cd` 命令进入指定 SDK 文件夹



```
root@1550f121ce14:/home/asriot# cd ASR582X_AliOS-Things/
root@1550f121ce14:/home/asriot/ASR582X_AliOS-Things#
```

图 A-13 进入 SDK 文件夹

## A.1.2 Linux 下 Docker 环境搭建

推荐使用 64 位 Ubuntu 系统

### A.1.2.1 安装 Docker 编译环境

```
#安装 docker 工具

sudo snap install docker

#获取 OpenHarmony docker 环境镜像

sudo docker pull iotasrmicro/asriot_buildenv:0.5

#进入 SDK 源码目录，创建 Docker 环境

sudo docker run --name harmony_docker_env -it -v $(pwd):/home/asriot
iotasrmicro/asriot_buildenv:0.5
```

#### a) 安装 Docker

在 Ubuntu 下执行如下命令安装 Docker: `sudo snap install docker`

```
jinghong@jinghong-virtual-machine:~/Documents/ASRCode/openharmony3.0_asr$ sudo snap install docker
[sudo] password for jinghong:
docker 20.10.14 from Canonical ✓ installed
```

图 A-14 安装 Docker

#### b) 获取 OpenHarmony 的 Docker 镜像

在 Ubuntu 下执行: `sudo docker pull iotasrmicro/asriot_buildenv:0.5`

```
gongjh@gongjh:~/Documents/ASRCode$ sudo docker pull iotasrmicro/asriot_buildenv:0.5
0.5: Pulling from iotasrmicro/asriot_buildenv
a70d879fa598: Pull complete
c4394a92d1f8: Pull complete
10e6159c56c0: Pull complete
76797d44d710: Pull complete
8491116844f7: Pull complete
7f855563119a: Pull complete
2a121927eacf: Pull complete
d77de761e987: Pull complete
8c19f595398a: Pull complete
Digest: sha256:a90121fc046e815192b17d9515ce07f8076f3a434753c49cccf4ca37d0d2e167
Status: Downloaded newer image for iotasrmicro/asriot_buildenv:0.5
docker.io/iotasrmicro/asriot_buildenv:0.5
```

图 A-15 获取 Openharmony Docker 镜像

#### c) 进入 SDK 源码目录执行如下命令，进入 Docker 构建环境

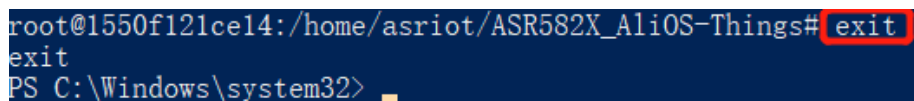
在 Ubuntu 下执行: `sudo docker run --name asr_build_env -it -v $(pwd):/home/asriot iotasrmicro/asriot_buildenv:0.5`

```
gongjh@gongjh:~/Documents/ASRCode$ sudo docker run --name asr_build_env -it -v $(pwd):/home/asriot iotasrmicro/asriot_buildenv:0.5
[sudo] password for gongjh:
root@bfb472ffb589:/home/asriot#
```

图 A-16 进入 Docker 构建环境

## A.2 退出 Docker 环境

如果想要退出当前 Docker 环境，使用如下命令：`exit`

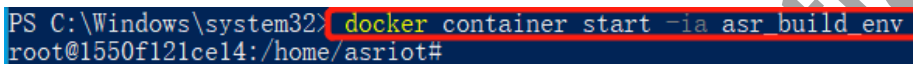


```
root@1550f121ce14:/home/asriot/ASR582X_AliOS-Things# exit
exit
PS C:\Windows\system32>
```

图 A-17 退出当前 Docker 环境

## A.3 再次进入 Docker 环境

如果想要再次进入之前的 Docker 环境，在管理员 **PowerShell** 或 **Terminal** 终端输入如下指令：  
`docker container start -ia asr_build_env` 或 `sudo docker container start -ia asr_build_env`。



```
PS C:\Windows\system32> docker container start -ia asr_build_env
root@1550f121ce14:/home/asriot#
```

图 A-18 再次进入 Docker 环境