

以 5.1.RC1.alpha002 版本为例，相关安装包可通过文档中的链接下载。

5.1.RC1.alpha002 版本分设场景部署文档

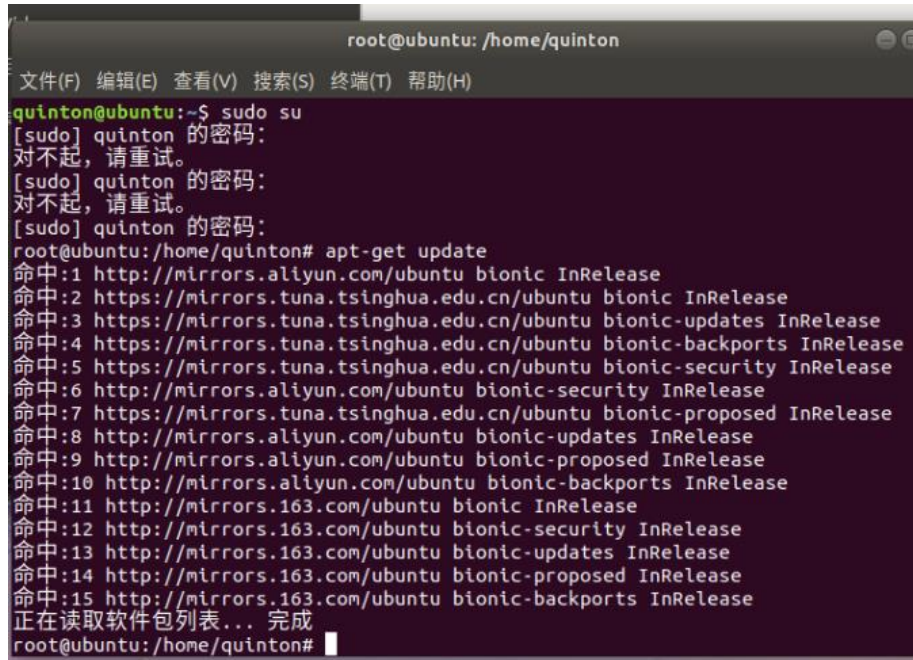
1、环境依赖安装

依赖列表

依赖	版本要求
Python	CANN支持Python3.7.x (3.7.0~3.7.11) 、Python3.8.x (3.8.0~3.8.11) 。
cmake	>=3.5.1
make	-
gcc	<ul style="list-style-type: none">离线推理场景要求4.8.5版本及以上gcc。在线推理、训练、Ascend Graph开发场景要求7.3.0版本及以上gcc，若gcc版本低于7.3.0，可参考12.2 安装7.3.0版本gcc进行安装。
g++	
libsqlite3-dev openssl libssl-dev libffi-dev unzip pciutils net-tools libblas-dev gfortran libblas3 liblapack-dev liblapack3 libopenblas-dev	
numpy	>=1.14.3
decorator	>=4.4.0
sympy	>=1.4
cffi	>=1.12.3
protobuf	>=3.11.3
attrs pyyaml pathlib2 scipy requests psutil	无版本要求，安装的版本以pip源为准。

- 进入 root 用户，检查源是否可用

1 | `apt-get update`



```
root@ubuntu: /home/quinton
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
quinton@ubuntu: ~$ sudo su
[sudo] quinton 的密码:
对不起，请重试。
[sudo] quinton 的密码:
对不起，请重试。
[sudo] quinton 的密码:
root@ubuntu: /home/quinton# apt-get update
命中:1 http://mirrors.aliyun.com/ubuntu bionic InRelease
命中:2 https://mirrors.tuna.tsinghua.edu.cn/ubuntu bionic InRelease
命中:3 https://mirrors.tuna.tsinghua.edu.cn/ubuntu bionic-updates InRelease
命中:4 https://mirrors.tuna.tsinghua.edu.cn/ubuntu bionic-backports InRelease
命中:5 https://mirrors.tuna.tsinghua.edu.cn/ubuntu bionic-security InRelease
命中:6 http://mirrors.aliyun.com/ubuntu bionic-security InRelease
命中:7 https://mirrors.tuna.tsinghua.edu.cn/ubuntu bionic-proposed InRelease
命中:8 http://mirrors.aliyun.com/ubuntu bionic-updates InRelease
命中:9 http://mirrors.aliyun.com/ubuntu bionic-proposed InRelease
命中:10 http://mirrors.aliyun.com/ubuntu bionic-backports InRelease
命中:11 http://mirrors.163.com/ubuntu bionic InRelease
命中:12 http://mirrors.163.com/ubuntu bionic-security InRelease
命中:13 http://mirrors.163.com/ubuntu bionic-updates InRelease
命中:14 http://mirrors.163.com/ubuntu bionic-proposed InRelease
命中:15 http://mirrors.163.com/ubuntu bionic-backports InRelease
正在读取软件包列表... 完成
root@ubuntu: /home/quinton#
```

- root 用户下执行如下安装命令

`apt-get install -y gcc g++ make cmake zlib1g zlib1g-dev openssl libsqlite3-dev libssl-dev libffi-dev unzip pciutils net-tools libblas-dev gfortran libblas3 libopenblas-dev`
安装完成

```
root@ubuntu: /home/quinton
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
nu)
update-alternatives: 使用 /usr/lib/x86_64-linux-gnu/openblas/liblapack.so.3 来在
自动模式中提供 /usr/lib/x86_64-linux-gnu/liblapack.so.3 (liblapack.so.3-x86_64-l
inux-gnu)
正在设置 libjsoncpp1:amd64 (1.7.4-3) ...
正在设置 gfortran (4:7.4.0-1ubuntu2.3) ...
update-alternatives: 使用 /usr/bin/gfortran 来在自动模式中提供 /usr/bin/f95 (f95
)
update-alternatives: 使用 /usr/bin/gfortran 来在自动模式中提供 /usr/bin/f77 (f77
)
正在设置 libblas-dev:amd64 (3.7.1-4ubuntu1) ...
update-alternatives: 使用 /usr/lib/x86_64-linux-gnu/blas/libblas.so 来在自动模式
中提供 /usr/lib/x86_64-linux-gnu/libblas.so (libblas.so-x86_64-linux-gnu)
正在设置 libopenblas-dev:amd64 (0.2.20+ds-4) ...
update-alternatives: 使用 /usr/lib/x86_64-linux-gnu/openblas/libblas.so 来在自动
模式中提供 /usr/lib/x86_64-linux-gnu/libblas.so (libblas.so-x86_64-linux-gnu)
update-alternatives: 使用 /usr/lib/x86_64-linux-gnu/openblas/liblapack.so 来在自
动模式中提供 /usr/lib/x86_64-linux-gnu/liblapack.so (liblapack.so-x86_64-linux-g
nu)
正在设置 cmake (3.10.2-1ubuntu2.18.04.2) ...
正在处理用于 libc-bin (2.27-3ubuntu1.4) 的触发器 ...
正在处理用于 man-db (2.8.3-2ubuntu0.1) 的触发器 ...
正在处理用于 install-info (6.5.0.dfsg.1-2) 的触发器 ...
root@ubuntu: /home/quinton#
```

- 检查系统是否安装满足版本要求的 python 开发环境

```
1 | python3 --version
```

如果返回信息满足 *python* 版本要求 (3.7.0~3.7.11), 则直接进入下一步, 否则可参考如下方式安装 *python3.7.5*

- a.使用 *wget* 下载 *python3.7.5* 源码包, 可以下载到安装环境的任意目录

```
1 | wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
```

```
root@ubuntu: /home/quinton
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
update-alternatives: 使用 /usr/lib/x86_64-linux-gnu/openblas/libblas.so 来在自动
模式中提供 /usr/lib/x86_64-linux-gnu/libblas.so (libblas.so-x86_64-linux-gnu)
update-alternatives: 使用 /usr/lib/x86_64-linux-gnu/openblas/liblapack.so 来在自
动模式中提供 /usr/lib/x86_64-linux-gnu/liblapack.so (liblapack.so-x86_64-linux-g
nu)
正在设置 cmake (3.10.2-1ubuntu2.18.04.2) ...
正在处理用于 libc-bin (2.27-3ubuntu1.4) 的触发器 ...
正在处理用于 man-db (2.8.3-2ubuntu0.1) 的触发器 ...
正在处理用于 install-info (6.5.0.dfsg.1-2) 的触发器 ...
root@ubuntu:/home/quinton# python3 --version
Python 3.6.9
root@ubuntu:/home/quinton# wget https://www.python.org/ftp/python/3.7.5/Python-3
.7.5.tgz
--2022-01-28 00:22:18-- https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tg
z
正在解析主机 www.python.org (www.python.org)... 151.101.76.223, 2a04:4e42:12::22
3
正在连接 www.python.org (www.python.org)|151.101.76.223|:443... 已连接。
已发出 HTTP 请求, 正在等待回应... 200 OK
长度: 23126230 (22M) [application/octet-stream]
正在保存至: "Python-3.7.5.tgz"

Python-3.7.5.tgz      12%[=>                ] 2.72M  2.69MB/s
Python-3.7.5.tgz      17%[==>                ] 3.80M  1.38MB/s
```

b.进入下载后的目录，解压源码包

```
1 | tar -zxvf Python-3.7.5.tgz
```

```
root@ubuntu: /home/quinton
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
Python-3.7.5/Objects/clinic/bytearrayobject.c.h
Python-3.7.5/Objects/clinic/enumobject.c.h
Python-3.7.5/Objects/clinic/bytesobject.c.h
Python-3.7.5/Objects/clinic/floatobject.c.h
Python-3.7.5/Objects/clinic/funcobject.c.h
Python-3.7.5/Objects/clinic/longobject.c.h
Python-3.7.5/Objects/clinic/dictobject.c.h
Python-3.7.5/Objects/clinic/structseq.c.h
Python-3.7.5/Objects/clinic/tupleobject.c.h
Python-3.7.5/Objects/clinic/moduleobject.c.h
Python-3.7.5/Objects/clinic/odictobject.c.h
Python-3.7.5/Objects/bytearrayobject.c
Python-3.7.5/Objects/typeobject.c
Python-3.7.5/Objects/lnotab_notes.txt
Python-3.7.5/Objects/methodobject.c
Python-3.7.5/Objects/tupleobject.c
Python-3.7.5/Objects/obmalloc.c
Python-3.7.5/Objects/object.c
Python-3.7.5/Objects/abstract.c
Python-3.7.5/Objects/listobject.c
Python-3.7.5/Objects/bytes_methods.c
Python-3.7.5/Objects/dictnotes.txt
Python-3.7.5/Objects/typeslots.inc
root@ubuntu:/home/quinton#
```

c.进入解压后的文件夹，执行配置、编译和安装命令

```
# 其中"--prefix"参数用于指定 python 安装路径, 可根据实际情况进行修改
cd Python-3.7.5
./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --
enable-shared
make
```

sudo make install

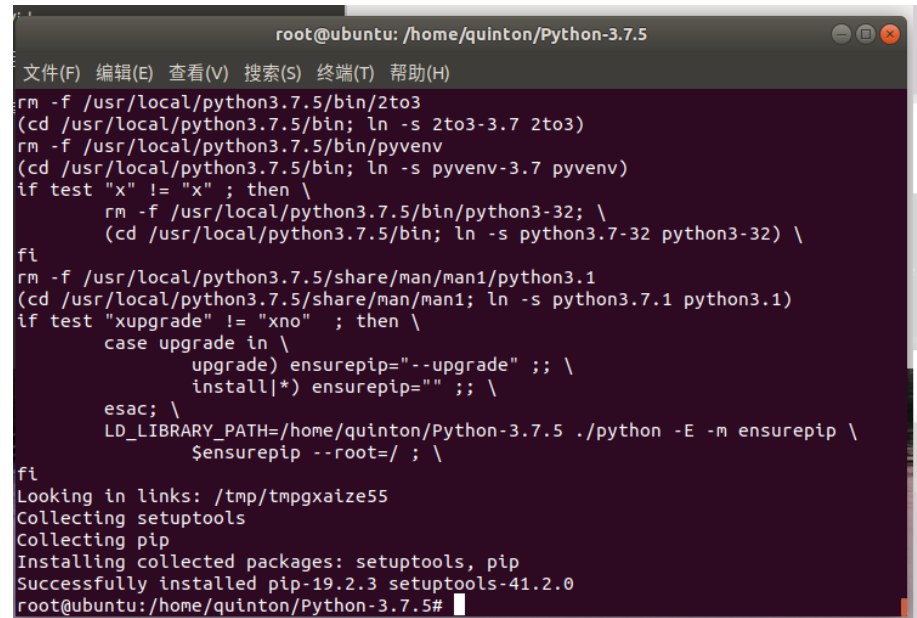
其中"--prefix"参数用于指定 python 安装路径, 可根据实际情况进行修改

cd Python-3.7.5

./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared

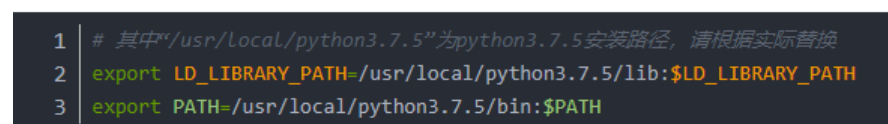
make

sudo make install

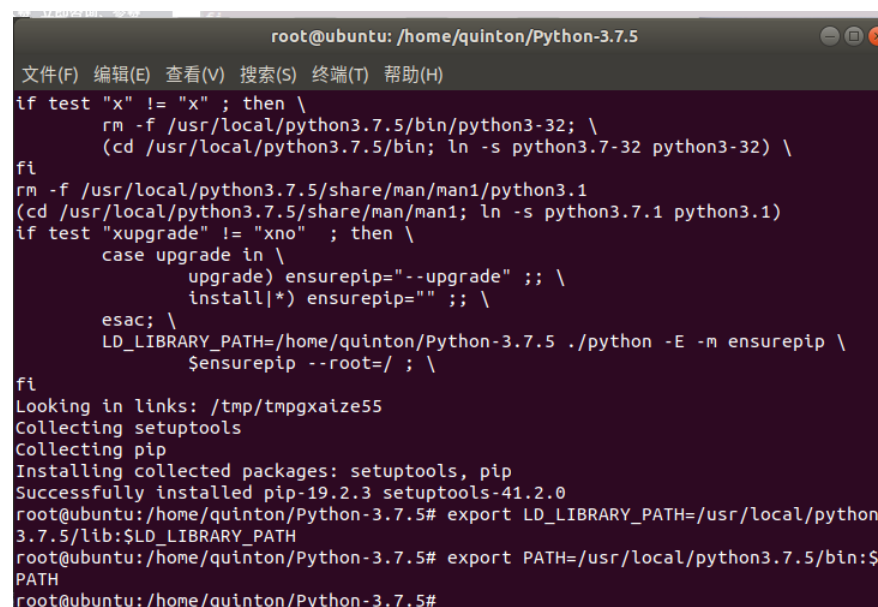


```
root@ubuntu: /home/quinton/Python-3.7.5
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
rm -f /usr/local/python3.7.5/bin/2to3
(cd /usr/local/python3.7.5/bin; ln -s 2to3-3.7 2to3)
rm -f /usr/local/python3.7.5/bin/pyvenv
(cd /usr/local/python3.7.5/bin; ln -s pyvenv-3.7 pyvenv)
if test "x" != "x" ; then \
    rm -f /usr/local/python3.7.5/bin/python3-32; \
    (cd /usr/local/python3.7.5/bin; ln -s python3.7-32 python3-32) \
fi
rm -f /usr/local/python3.7.5/share/man/man1/python3.1
(cd /usr/local/python3.7.5/share/man/man1; ln -s python3.7.1 python3.1)
if test "xupgrade" != "xno" ; then \
    case upgrade in \
        upgrade) ensurepip="--upgrade" ;; \
        install|*) ensurepip="" ;; \
    esac; \
    LD_LIBRARY_PATH=/home/quinton/Python-3.7.5 ./python -E -m ensurepip \
    $ensurepip --root=/ ; \
fi
Looking in links: /tmp/tmpgxaize55
Collecting setuptools
Collecting pip
Installing collected packages: setuptools, pip
Successfully installed pip-19.2.3 setuptools-41.2.0
root@ubuntu:/home/quinton/Python-3.7.5#
```

d. 设置 python3.7.5 环境变量



```
1 # 其中"/usr/local/python3.7.5"为python3.7.5安装路径, 请根据实际替换
2 export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
3 export PATH=/usr/local/python3.7.5/bin:$PATH
```



```
root@ubuntu: /home/quinton/Python-3.7.5
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
if test "x" != "x" ; then \
    rm -f /usr/local/python3.7.5/bin/python3-32; \
    (cd /usr/local/python3.7.5/bin; ln -s python3.7-32 python3-32) \
fi
rm -f /usr/local/python3.7.5/share/man/man1/python3.1
(cd /usr/local/python3.7.5/share/man/man1; ln -s python3.7.1 python3.1)
if test "xupgrade" != "xno" ; then \
    case upgrade in \
        upgrade) ensurepip="--upgrade" ;; \
        install|*) ensurepip="" ;; \
    esac; \
    LD_LIBRARY_PATH=/home/quinton/Python-3.7.5 ./python -E -m ensurepip \
    $ensurepip --root=/ ; \
fi
Looking in links: /tmp/tmpgxaize55
Collecting setuptools
Collecting pip
Installing collected packages: setuptools, pip
Successfully installed pip-19.2.3 setuptools-41.2.0
root@ubuntu:/home/quinton/Python-3.7.5# export LD_LIBRARY_PATH=/usr/local/python
3.7.5/lib:$LD_LIBRARY_PATH
root@ubuntu:/home/quinton/Python-3.7.5# export PATH=/usr/local/python3.7.5/bin:$
PATH
root@ubuntu:/home/quinton/Python-3.7.5#
```

e. 检查是否安装成功

```
root@ubuntu: /home/quinton/Python-3.7.5
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
(cd /usr/local/python3.7.5/share/man/man1; ln -s python3.7.1 python3.1)
if test "xupgrade" != "xno" ; then \
    case upgrade in \
        upgrade) ensurepip="--upgrade" ;; \
        install|*) ensurepip="" ;; \
        esac; \
    LD_LIBRARY_PATH=/home/quinton/Python-3.7.5 ./python -E -m ensurepip \
        $ensurepip --root=/ ; \
fi
Looking in links: /tmp/tmpgxaize55
Collecting setuptools
Collecting pip
Installing collected packages: setuptools, pip
Successfully installed pip-19.2.3 setuptools-41.2.0
root@ubuntu: /home/quinton/Python-3.7.5# export LD_LIBRARY_PATH=/usr/local/python
3.7.5/lib:$LD_LIBRARY_PATH
root@ubuntu: /home/quinton/Python-3.7.5# export PATH=/usr/local/python3.7.5/bin:$
PATH
root@ubuntu: /home/quinton/Python-3.7.5# vi /etc/use_private_python.info
root@ubuntu: /home/quinton/Python-3.7.5# vi /etc/use_private_python.info
root@ubuntu: /home/quinton/Python-3.7.5# python3 --version
Python 3.7.5
root@ubuntu: /home/quinton/Python-3.7.5#
root@ubuntu: /home/quinton/Python-3.7.5#
```

- 为后续安装 CANN 软件包、运行 CANN 软件环境变量设置脚本时能够自动配置 python3.7.5 环境变量，需提前创建好文件“use_private_python.info”

root 用户执行如下命令创建文件

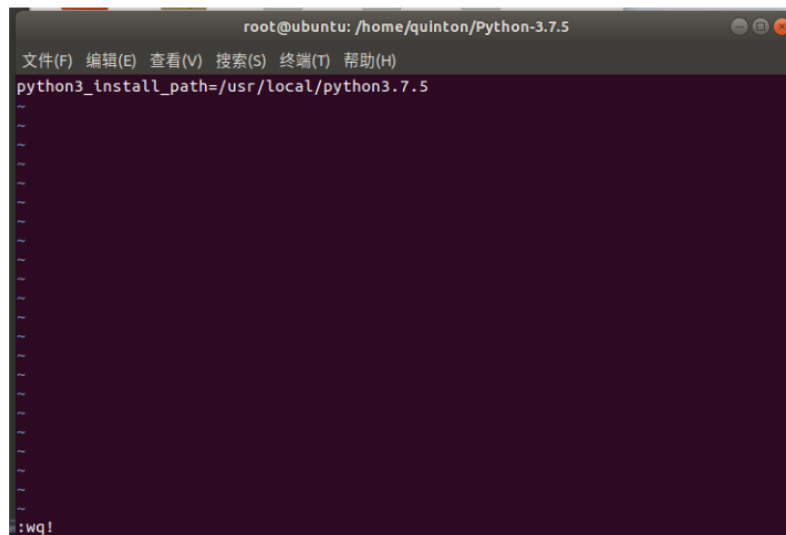
```
1 | vi /etc/use_private_python.info
```

```
root@ubuntu: /home/quinton/Python-3.7.5
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

/etc/use_private_python.info" [New File]
```

添加此内容，保存退出

```
1 | python3_install_path=/usr/local/python3.7.5
```



- 使用 pip 命令安装依赖

如下命令如果使用非 root 用户安装，需要在安装命令后加上—user

例如：pip3 install attrs --user，安装命令可在任意路径下执行。

```
1 # 需要进入root用户
2
3 # 更新pip命令
4 pip3 install --upgrade pip
5 # 安装依赖
6 pip3 install attrs
7 pip3 install numpy==1.17.2
8 pip3 install decorator
9 pip3 install sympy
10 pip3 install cffi
11 pip3 install pyyaml
12 pip3 install pathlib2
13 pip3 install psutil
14 pip3 install protobuf
15 pip3 install scipy
16 pip3 install requests
```

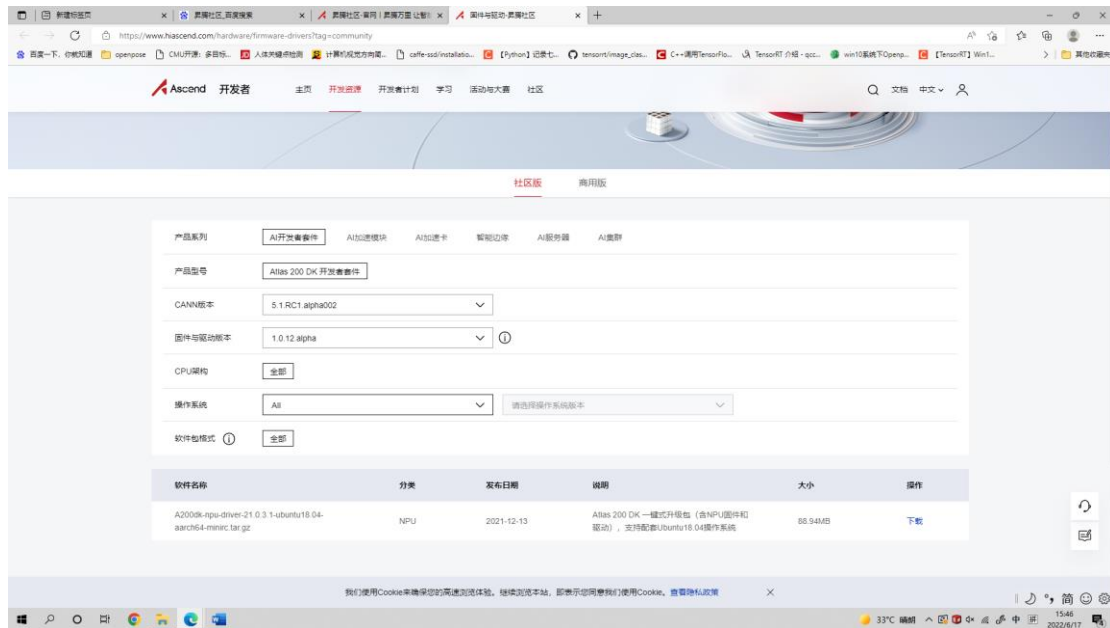
安装完成

2、制卡环节

软件包准备 获取开发者板驱动与运行包、Ubuntu 操作系统镜像

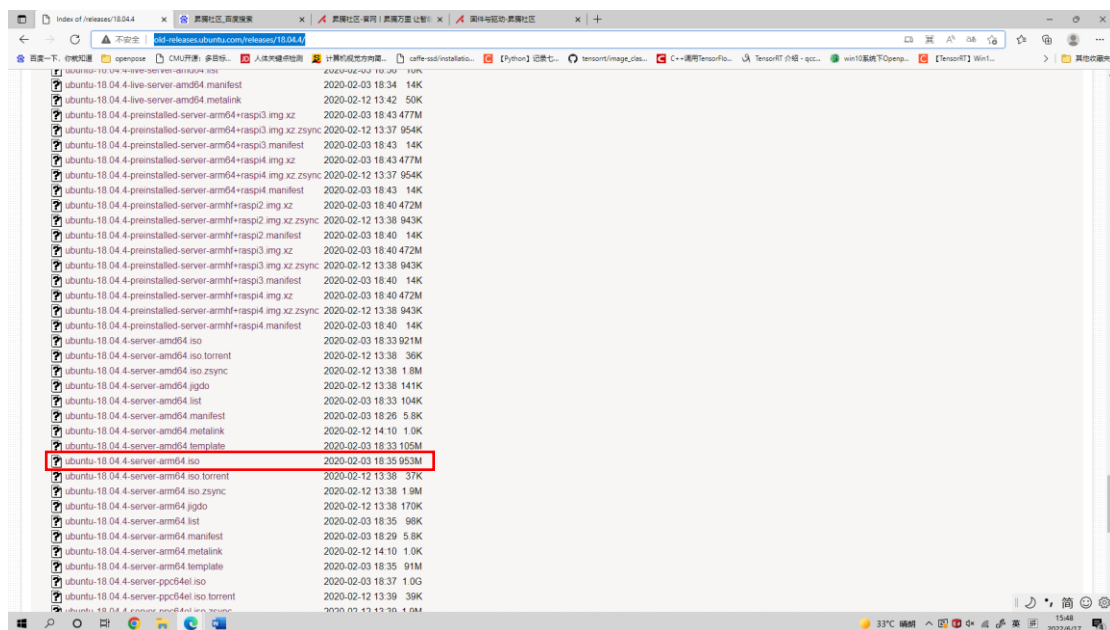
驱动包对应版本如下图所示：

链接：A200dk-npu-driver-21.0.3.1-ubuntu18.04-aarch64-minirc.tar.gz



Ubuntu 操作系统镜像选择对应版本为 18.04.4

链接：[ubuntu-18.04.4-server-arm64.iso](https://releases.ubuntu.com/18.04.4/)



将 SD 卡放入读卡器，并将读卡器与 Ubuntu 服务器的 USB 接口连接

Ubuntu 服务器中安装 qemu-user-static、binfmt-support、yaml、squashfs-tools 与交叉编译器

进入 root 用户

sudo su

更新源

apt-get update

安装 python 相关依赖

pip3 install pyyaml

安装 qemu-user-static、binfmt-support、yaml、squashfs-tools 与交叉编译器
apt-get install qemu-user-static binfmt-support python3-yaml squashfs-tools gcc-aarch64-
linux-gnu g++-aarch64-linux-gnu

安装完成

```
root@ubuntu: /home/quinton
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
正在设置 libgomp1-arm64-cross (8.4.0-1ubuntu1~18.04cross2) ...
正在设置 linux-libc-dev-arm64-cross (4.15.0-35.38cross1.1) ...
正在设置 cpp-aarch64-linux-gnu (4:7.4.0-1ubuntu2.3) ...
正在设置 qemu-user-static (1:2.11+dfsg-1ubuntu7.38) ...
正在设置 libgcc1-arm64-cross (1:8.4.0-1ubuntu1~18.04cross2) ...
正在设置 liblsan0-arm64-cross (8.4.0-1ubuntu1~18.04cross2) ...
正在设置 libasan4-arm64-cross (7.5.0-3ubuntu1~18.04cross1) ...
正在设置 libatomic1-arm64-cross (8.4.0-1ubuntu1~18.04cross2) ...
正在设置 libstdc++6-arm64-cross (8.4.0-1ubuntu1~18.04cross2) ...
正在设置 libubsan0-arm64-cross (7.5.0-3ubuntu1~18.04cross1) ...
正在设置 libtsan0-arm64-cross (8.4.0-1ubuntu1~18.04cross2) ...
正在设置 libitm1-arm64-cross (8.4.0-1ubuntu1~18.04cross2) ...
正在设置 libc6-dev-arm64-cross (2.27-3ubuntu1cross1.1) ...
正在设置 libgcc-7-dev-arm64-cross (7.5.0-3ubuntu1~18.04cross1) ...
正在设置 libstdc++-7-dev-arm64-cross (7.5.0-3ubuntu1~18.04cross1) ...
正在设置 gcc-7-aarch64-linux-gnu (7.5.0-3ubuntu1~18.04cross1) ...
正在设置 gcc-aarch64-linux-gnu (4:7.4.0-1ubuntu2.3) ...
正在设置 g++-7-aarch64-linux-gnu (7.5.0-3ubuntu1~18.04cross1) ...
正在设置 g++-aarch64-linux-gnu (4:7.4.0-1ubuntu2.3) ...
正在处理用于 systemd (237-3ubuntu10.53) 的触发器 ...
正在处理用于 man-db (2.8.3-2ubuntu0.1) 的触发器 ...
正在处理用于 ureadahead (0.100.0-21) 的触发器 ...
正在处理用于 libc-bin (2.27-3ubuntu1.4) 的触发器 ...
root@ubuntu: /home/quinton#
```

创建制卡工作目录

mkdir /home/ascend/mksd

制卡目录可任意指定

#ascend 为自己的用户名

将软件包准备获取的 Ubuntu 操作系统镜像包、开发者板所有驱动包上传到制卡工作目录
在制卡工作目录下获取制卡脚本。可点击链接直接下载或在 Ubuntu 内输入命令下载

[make_sd_card.py](#)

[make_ubuntu_sd.sh](#)

下载制卡入口脚本“make_sd_card.py”。

从 gitee 下载:

wget https://gitee.com/ascend/tools/raw/master/makesd/generic_script/make_sd_card.py

从 github 下载:

wget

https://raw.githubusercontent.com/Ascend/tools/master/makesd/generic_script/make_sd_card.py

下载制作 SD 卡操作系统的脚本“make_ubuntu_sd.sh”。

从 gitee 下载：

wget https://gitee.com/ascend/tools/raw/master/makesd/generic_script/make_ubuntu_sd.sh

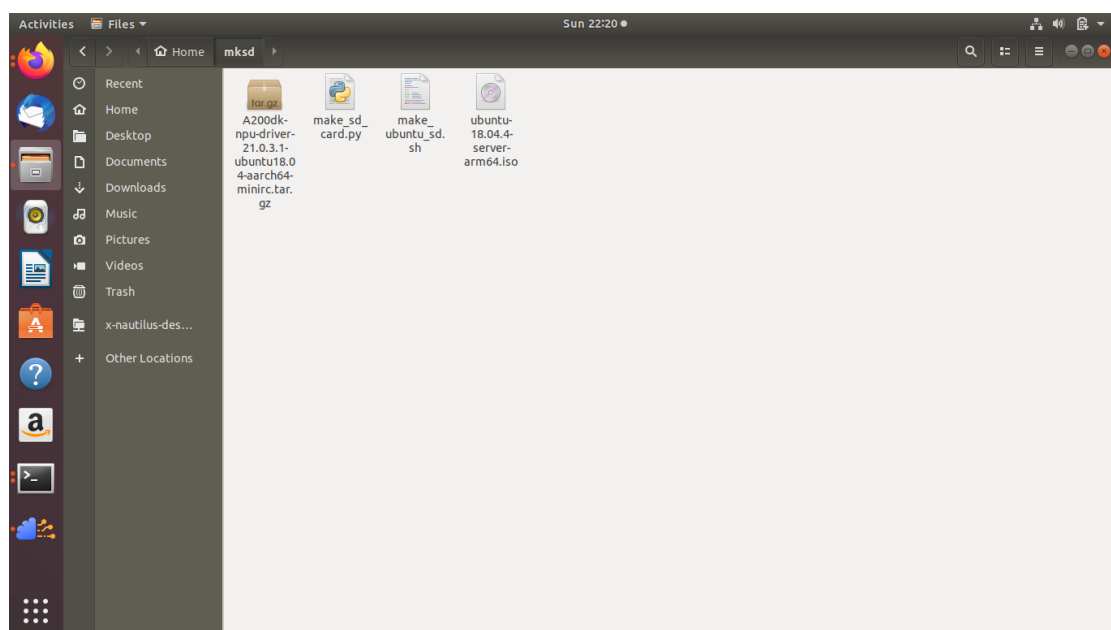
从 github 下载：

wget

https://raw.githubusercontent.com/Ascend/tools/master/makesd/generic_script/make_ubuntu_sd.sh

下载完成

mkzd 文件夹内容如下：



执行制卡脚本。

以 root 用户执行如下命令查找 SD 卡所在的 USB 设备名称。

`fdisk -l`

例如，SD 卡所在 USB 设备名称为“/dev/sda”，可通过插拔 SD 卡或者按容量推断的方式确定设备名称。

图中的 sd 卡为/dev/sdb

运行 SD 制卡脚本“make_sd_card.py”。

`python3 make_sd_card.py local /dev/sda1`

“local”表示使用本地方式制作 SD 卡。

“/dev/sda”为 SD 卡所在的 USB 设备名称。

正在制卡

制卡完成

拔出 sd 卡，插入开发板并上电，启动开发板

3、开发环境部署（PC 端）

此种场景下，用户 PC 机（Ubuntu x86 操作系统）作为开发环境，进行程序的开发、编译，QA500DK-Atlas 200 作为运行环境。

下面分别介绍开发环境与运行环境中 CANN 软件安装流程。

开发环境中 CANN 软件安装

1) 准备软件包。

请参见[下载链接](#)下载配套驱动版本的开发套件包“Ascend-cann-toolkit_{version}_linux-x86_64.run”与“Ascend-cann-toolkit_{version}_linux-aarch64.run”。

驱动与 CANN 版本的配套关系请参见[版本配套说明（Gitee）](#)或[版本配套说明（Github）](#)。

或者点击已下链接直接下载：

[Ascend-cann-toolkit_5.1.RC1.alpha002_linux-x86_64.run](#)

[Ascend-cann-toolkit_5.1.RC1.alpha002_linux-aarch64.run](#)

2) 准备参考资料。

请从[开发者文档](#)，获取配套版本的《CANN 软件安装指南》。

3) 准备安装及运行用户。

请参见《CANN 软件安装指南》的“安装开发环境 > 准备安装及运行用户”准备需要进行 CANN 软件安装的用户。

4) 安装 OS 依赖。

已在步骤 1 完成

5) 安装开发套件包。

a. 以安装用户将开发套件包上传到开发环境任意路径，并进入套件包所在路径。

b. 执行如下命令为安装包增加可执行权限。

```
chmod +x *.run
```

c. 执行如下校验安装包的一致性和完整性。

```
./Ascend-cann-toolkit_{version}_linux-x86_64.run --check
```

```
./Ascend-cann-toolkit_{version}_linux-aarch64.run --check
```

e. 执行以下命令安装软件。

```
f. ./Ascend-cann-toolkit_{version}_linux-x86_64.run --install --chip=Ascend310-minirc
```

```
./Ascend-cann-toolkit_{version}_linux-aarch64.run --install --chip=Ascend310-minirc
```

--chip=Ascend310-minirc（可选）：指定芯片型号为 Ascend310 Soc 芯片（RC 模式启动，作为主控 CPU）。配置了此参数，则会只部署 Ascend310RC 形态的 AI CPU 软件包。

以上命令示例使用默认路径进行安装，若安装用户为 root 用户，则默认路径为“/usr/local/Ascend”；若安装用户为非 root 用户，则默认路径为“\$HOME/Ascend”。

开发者也可以通过--install-path=<path>指定安装路径，安装支持的其他详细参数说明可参见《CANN 软件安装指南》的“参考信息 > 参数说明”。

6) 配置交叉编译环境。

分设场景下，开发环境架构为“x86_64”，运行环境架构为“aarch64”，所以在开发环境中安装交叉编译器进行应用程序的交叉编译。

请使用 CANN 软件包安装用户，在开发环境执行 **aarch64-linux-gnu-g++ --version** 命令检查交叉编译器是否安装，若已经安装则可以忽略；否则请参考以下命令进行安装（以下命令仅为示例，请用户根据实际情况替换）：

```
sudo apt-get install g++-aarch64-linux-gnu
```

7) 配置环境变量。

CANN 软件提供进程级环境变量设置脚本，供用户在进程中引用，以自动完成环境变量设置。用户进程结束后自动失效。示例如下（以 HwHiAiUser 用户默认安装路径为例）：

```
./home/HwHiAiUser/Ascend/ascend-toolkit/set_env.sh
```

用户也可以通过修改~/.bashrc 文件方式设置永久环境变量，操作如下：

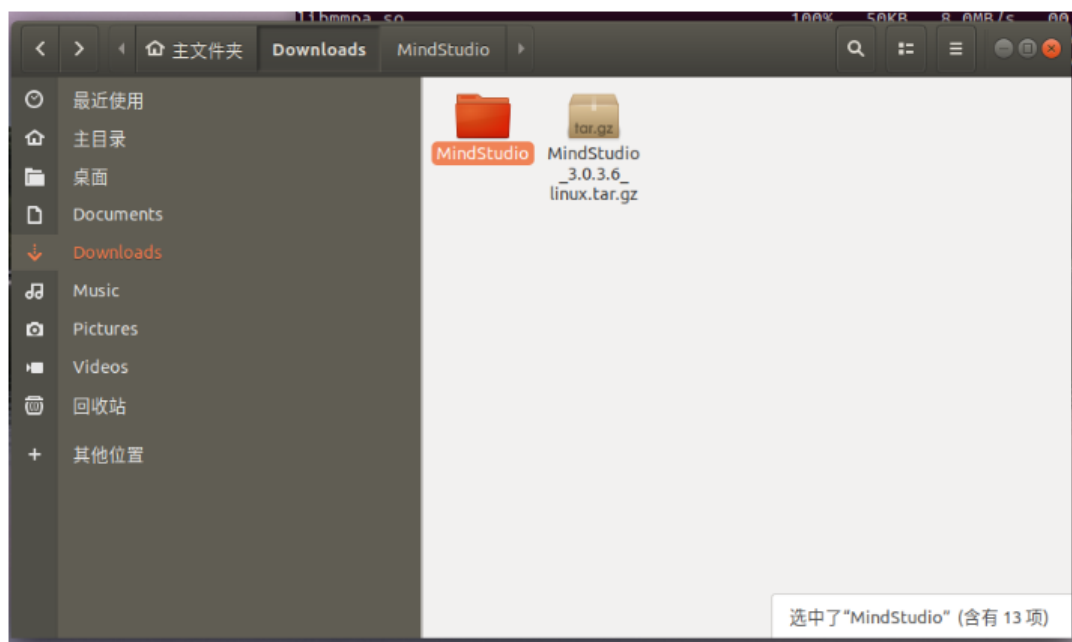
- 以运行用户在任意目录下执行 **vi ~/.bashrc** 命令，打开.bashrc 文件，在文件最后一行后面添加上述内容。
- 执行:wq!命令保存文件并退出。
- 执行 **source ~/.bashrc** 命令使其立即生效。

安装 MindStudio

下载包链接：[MindStudio_5.0.RC1_linux.tar.gz](#)

解压 MindStudio 软件包

```
1 | tar -zxvf MindStudio_{version}_linux.tar.gz
```



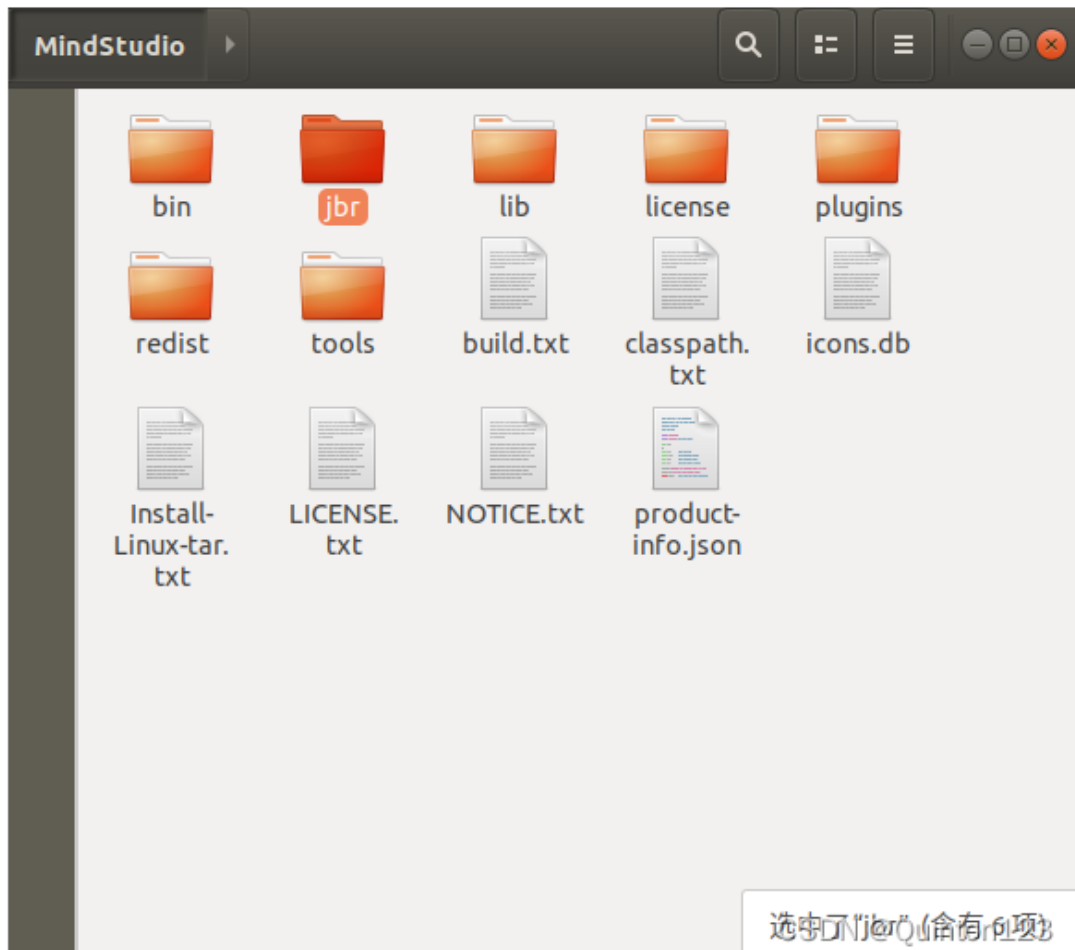
解压 jbr 至 MindStudio 安装根目录

下载 jbr

```
wget https://cache-redirector.jetbrains.com/intellij-jbr/jbr_dcevm-11_0_10-linux-x64-b1341.35.tar.gz
```

解压

```
tar -zxvf jbr_dcevm-11_0_10-linux-x64-b1341.35.tar.gz
```

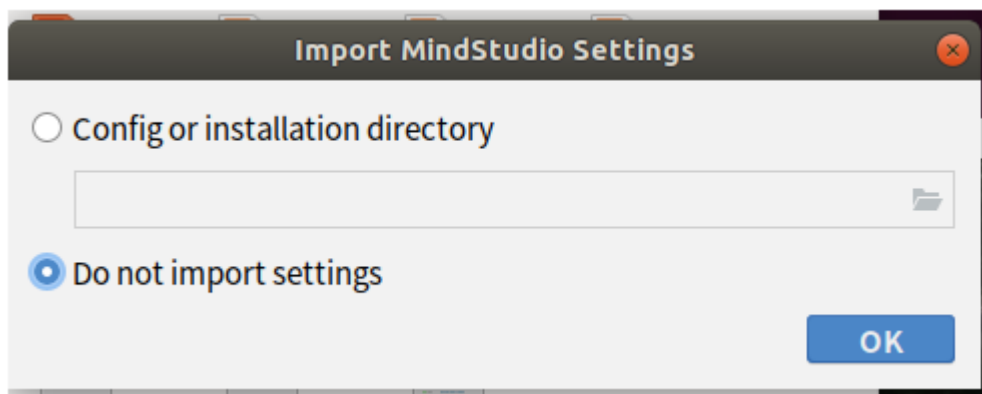


执行启动脚本

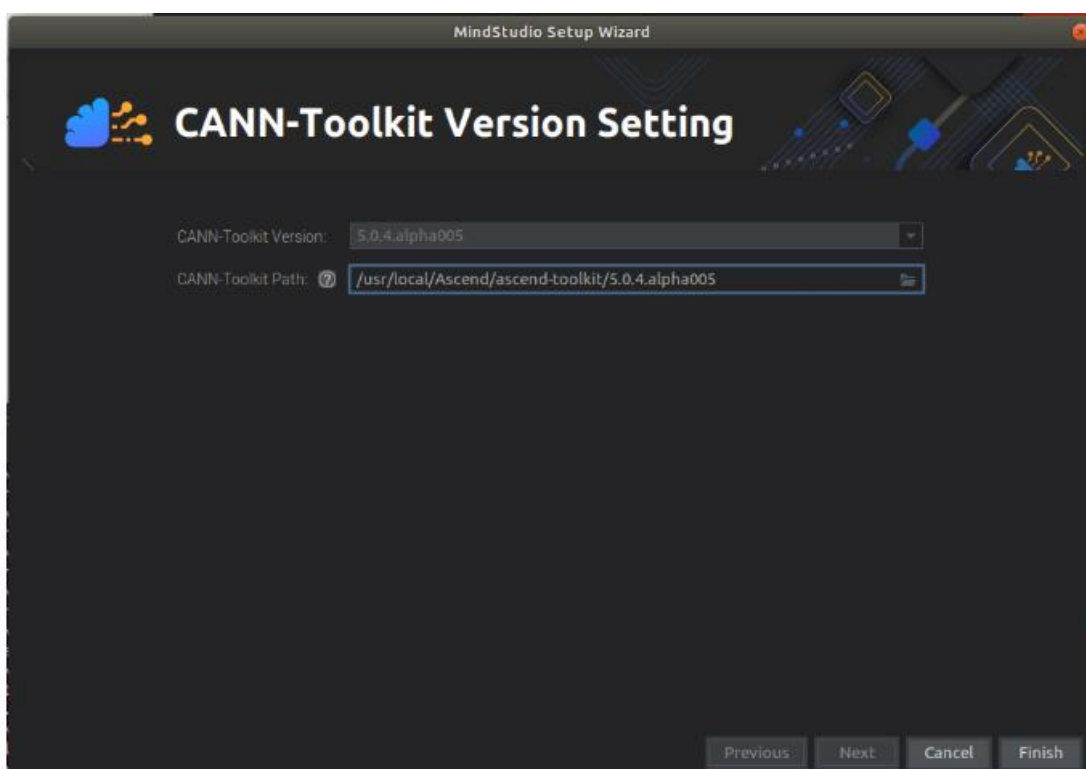
进入软件包解压后的 MindStudio/bin 目录，执行如下命令

```
1 | cd MindStudio/bin
2 | ./MindStudio.sh
```

- Do not import settings: 不导入设置，若选择该选项，则创建新的配置文件，默认为该选项



- 选择好 cann 软件安装的路径，点击 finish



配置编译环境

使用 MindStudio 安装用户,在 MindStudio 安装服务器执行 aarch64-linux-gnu-g++ --version 命令检查是否安装，若已经安装则可以忽略，否则执行如下安装命令：

```
1 | sudo apt-get install -y g++-aarch64-linux-gnu
```


4、运行环境部署（开发板）

1. 准备软件包。

直接下载链接：[Ascend-cann-nnrt_5.1.RC1.alpha002_linux-aarch64.run](#)

或参见[下载链接](#)下载配套驱动版本的离线推理引擎包“Ascend-cann-nnrt_{software version}_linux-aarch64.run”。

驱动与 CANN 版本的配套关系请参见[版本配套说明（Gitee）](#)或[版本配套说明（Github）](#)。

2. 在 QA500DK-Atlas 200 上安装离线推理引擎包。

请以制卡时创建的 **HwHiAiUser** 用户作为 Atlas 200 DK 上 CANN 软件的安装用户，安装步骤请参见《CANN 软件安装指南》的“安装运行环境（nnrt 软件，在物理机/虚拟机安装）> 安装离线推理引擎包”。

a. 以 HwHiAiUser 用户将离线推理引擎包上传到 Atlas 200 DK 任意目录。

b. 执行如下命令为安装包增加可执行权限。

```
chmod +x *.run
```

其中 ***.run** 表示软件包名，请根据实际包名进行替换。

c. 执行如下校验安装包的一致性和完整性。

```
./Ascend-cann-nnrt_{software version}_linux-aarch64.run --check
```

d. 执行如下命令进行离线推理引擎包的安装。

```
./Ascend-cann-nnrt_{software version}_linux-aarch64.run --install --chip=Ascend310-minirc
```

--chip=Ascend310-minirc（可选）：指定芯片型号为 Ascend310 Soc 芯片（RC 模式启动，作为主控 CPU）。

配置了此参数，则会只部署 Ascend310RC 形态的 AI CPU 软件包。

以上命令为使用默认路径进行安装的示例，默认安装路径为“\$HOME/Ascend”，安装后的 CANN 软件存储在“\$HOME/Ascend/nnrt/latest”中。

3. 配置环境变量。

CANN 软件提供进程级环境变量设置脚本，供用户在进程中引用，以自动完成环境变量设置。用户进程结束后自动失效。示例如下（以 HwHiAiUser 用户默认安装路径为例）：

```
. /home/HwHiAiUser/Ascend/nnrt/set_env.sh
```

用户也可以通过修改 `~/.bashrc` 文件方式设置永久环境变量，操作如下：

- 以运行用户在任意目录下执行 **vi ~/.bashrc** 命令，打开 **.bashrc** 文件，在文件最后一行后面添加上述内容。
- 执行 **:wq!** 命令保存文件并退出。
- 执行 **source ~/.bashrc** 命令使其立即生效。

5、第三方依赖安装指导(C++样例)

开发环境

请执行以下命令进行安装准备，**其中 CPU_ARCH 环境变量请根据运行环境 cpu 架构填写！**

...

以安装用户在开发环境任意目录下执行以下命令，打开.bashrc 文件。

vi ~/.bashrc

在文件最后一行后面添加如下内容。CPU_ARCH 环境变量请根据运行环境 cpu 架构填写，如 export CPU_ARCH=aarch64

export CPU_ARCH=aarch64

THIRDPART_PATH 需要按照运行环境安装路径设置，如运行环境为 arm，指定安装路径为 Ascend-arm，则需要设置为

export THIRDPART_PATH=\${HOME}/Ascend-arm/thirdpart/\${CPU_ARCH}

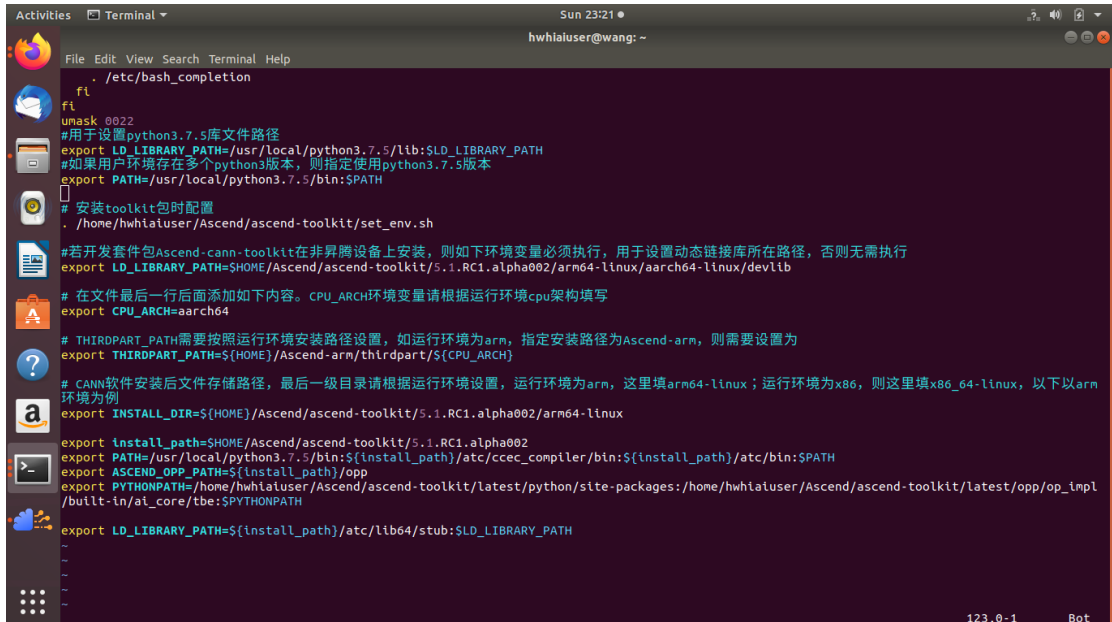
CANN 软件安装后文件存储路径，最后一级目录请根据运行环境设置，运行环境为 arm，这里填 arm64-linux；运行环境为 x86，则这里填 x86_64-linux，以下以 arm 环境为例

export INSTALL_DIR=\${HOME}/Ascend/ascend-toolkit/latest/arm64-linux

#此外，满足 atc 模型转换所需路径如下：

export LD_LIBRARY_PATH=\$HOME/Ascend/ascend-toolkit/latest/arm64-linux/aarch64-linux/devlib

环境配置如下：



```
fl
unask 0022
#用于设置python3.7.5库文件路径
export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
#如果用户环境存在多个python3版本，则指定使用python3.7.5版本
export PATH=/usr/local/python3.7.5/bin:$PATH
# 安装toolkit包时配置
./home/hwhialuser/Ascend/ascend-toolkit/set_env.sh
#若开发套件Ascend-cann-toolkit在非昇腾设备上安装，则如下环境变量必须执行，用于设置动态链接库所在路径，否则无需执行
export LD_LIBRARY_PATH=$HOME/Ascend/ascend-toolkit/5.1.RC1.alpha002/arm64-linux/aarch64-linux/devlib
# 在文件最后一行后面添加如下内容。CPU_ARCH环境变量请根据运行环境cpu架构填写
export CPU_ARCH=aarch64
# THIRDPART_PATH需要按照运行环境安装路径设置，如运行环境为arm，指定安装路径为Ascend-arm，则需要设置为
export THIRDPART_PATH=${HOME}/Ascend-arm/thirdpart/${CPU_ARCH}
# CANN软件安装后文件存储路径，最后一级目录请根据运行环境设置，运行环境为arm，这里填arm64-linux；运行环境为x86，则这里填x86_64-linux，以下以arm环境为例
export INSTALL_DIR=${HOME}/Ascend/ascend-toolkit/5.1.RC1.alpha002/arm64-linux
export install_path=$HOME/Ascend/ascend-toolkit/5.1.RC1.alpha002
export PATH=/usr/local/python3.7.5/bin:${install_path}/atc/cccec_compiler/bin:${install_path}/atc/bin:$PATH
export ASCEND_OPP_PATH=${install_path}/opp
export PYTHONPATH=/home/hwhialuser/Ascend/ascend-toolkit/latest/python/site-packages:/home/hwhialuser/Ascend/ascend-toolkit/latest/opp/op_impl/built-in/at_core/tbe:$PYTHONPATH
export LD_LIBRARY_PATH=${install_path}/atc/lib64/stub:$LD_LIBRARY_PATH
~
~
~
123,0-1 Bot
```

执行命令保存文件并退出。

:wq!

执行命令使其立即生效。

source ~/.bashrc

```
# 创建第三方依赖文件夹
mkdir -p ${THIRDPART_PATH}
# 拷贝公共文件到第三方依赖文件夹
cd $HOME
git clone https://gitee.com/ascend/samples.git
cp -r ${HOME}/samples/common ${THIRDPART_PATH}
...
```

如果运行环境是 200DK，还需要执行以下命令拷贝 media_mini 等动态库及相关头文件，满足摄像头样例编译需要。

```
...
mkdir -p ${INSTALL_DIR}/driver
sudo scp -r HwHiAiUser@X.X.X.X:/usr/lib64/libmedia_mini.so ${INSTALL_DIR}/driver/
sudo scp -r HwHiAiUser@X.X.X.X:/usr/lib64/libclog.so ${INSTALL_DIR}/driver/
sudo scp -r HwHiAiUser@X.X.X.X:/usr/lib64/libc_sec.so ${INSTALL_DIR}/driver/
sudo scp -r HwHiAiUser@X.X.X.X:/usr/lib64/libmma.so ${INSTALL_DIR}/driver/
sudo scp -r HwHiAiUser@X.X.X.X:/usr/local/Ascend/include/peripheral_api.h
${INSTALL_DIR}/driver/
```

如果运行环境是 200DK，还需要执行以下命令拷贝 media_mini 等动态库及相关头文件，满足摄像头样例编译需要。

```
...
mkdir -p ${INSTALL_DIR}/driver
sudo scp -r HwHiAiUser@X.X.X.X:/usr/lib64/libmedia_mini.so ${INSTALL_DIR}/driver/
sudo scp -r HwHiAiUser@X.X.X.X:/usr/lib64/libclog.so ${INSTALL_DIR}/driver/
sudo scp -r HwHiAiUser@X.X.X.X:/usr/lib64/libc_sec.so ${INSTALL_DIR}/driver/
sudo scp -r HwHiAiUser@X.X.X.X:/usr/lib64/libmma.so ${INSTALL_DIR}/driver/
sudo scp -r HwHiAiUser@X.X.X.X:/usr/local/Ascend/include/peripheral_api.h
${INSTALL_DIR}/driver/
```

运行环境

请执行以下命令进行安装准备

```
...
# 以安装用户在运行环境任意目录下执行以下命令，打开.bashrc 文件。
vi ~/.bashrc
# 在文件最后一行后面添加如下内容。CPU_ARCH 环境变量请根据运行环境 cpu 架构填写，如 export CPU_ARCH=aarch64
export CPU_ARCH=aarch64
export THIRDPART_PATH=${HOME}/Ascend/thirdpart/${CPU_ARCH} #代码编译时链接第三方库
export
LD_LIBRARY_PATH=${HOME}/Ascend/thirdpart/${CPU_ARCH}/lib:$LD_LIBRARY_PATH #运行时链接库文件
```

```

export INSTALL_DIR=${HOME}/Ascend/nnrt/latest #CANN 软件安装后文件存储路径
# 执行命令保存文件并退出。
:wq!
# 执行命令使其立即生效。
source ~/.bashrc
# 创建第三方依赖文件夹
mkdir -p ${THIRDPART_PATH}
# 拷贝相关数据，将开发环境中${THIRDPART_PATH}/common 上传至运行环境的
${THIRDPART_PATH}的路径下。
sudo scp -r HwHiAiUser@X.X.X.X:${THIRDPART_PATH}/common ${THIRDPART_PATH}
...

```

如果运行环境是 200DK，还需要执行以下命令拷贝 media_mini 等动态库及相关头文件，满足摄像头样例运行需要。

```

...

mkdir ${INSTALL_DIR}/driver
cp /usr/lib64/libmedia_mini.so ${INSTALL_DIR}/driver/
cp /usr/lib64/liblog.so ${INSTALL_DIR}/driver/
cp /usr/lib64/libc_sec.so ${INSTALL_DIR}/driver/
cp /usr/lib64/libmmpa.so ${INSTALL_DIR}/driver/
cp /usr/local/Ascend/include/peripheral_api.h ${INSTALL_DIR}/driver/

```

依赖安装

安装 opencv

由于适配不同运行环境，opencv 的安装存在差异较大，请选择以下适合的场景进行安装。如果代码中不涉及 opencv，则可以跳过此步骤。

- 开发环境架构为 X86，运行环境为 x86。
****开发环境和运行环境均****执行以下命令安装 opencv。

```

...

sudo apt-get install libopencv-dev
...

```

- 开发环境为 x86，运行环境为 arm。

由于源码安装交叉编译较为复杂，所以这里在运行环境上直接使用 apt 安装 opencv，安装完成后拷贝回开发环境即可。

1. ****运行环境****联网并执行以下命令进行安装

```

...

sudo apt-get install libopencv-dev
...

```

2. ****开发环境****执行以下命令拷贝对应 so

```

...

```

将 arm 下的 opencv 相关的 so 拷贝到 X86 的 aarch64-linux-gnu 目录，不会对本地 X86 环境本身使用产生任何问题。

```

cd /usr/lib/aarch64-linux-gnu
# 拷贝相关 so，其中 X.X.X.X 为运行环境 ip 地址。
sudo scp -r HwHiAiUser@X.X.X.X:/lib/aarch64-linux-gnu/* ./
sudo scp -r HwHiAiUser@X.X.X.X:/usr/lib/aarch64-linux-gnu/* ./
sudo scp -r HwHiAiUser@X.X.X.X:/usr/lib/*.so.* ./
# 拷贝 opencv 相关头文件。
sudo scp -r HwHiAiUser@X.X.X.X:/usr/include/opencv* /usr/include
...

```

安装 ffmpeg+acclite

开发环境执行以下命令源码安装 ffmpeg（apt 安装的 ffmpeg 版本较低，所以源码安装）并安装 acclite。如果代码中并没有使用 acclite 库相关功能及函数，可以跳过此步骤。

A. 下载并安装 ffmpeg。

```

...
# 下载 ffmpeg
cd ${HOME}
wget http://www.ffmpeg.org/releases/ffmpeg-4.1.3.tar.gz --no-check-certificate
tar -zxvf ffmpeg-4.1.3.tar.gz
cd ffmpeg-4.1.3
...

```

B. 安装 ffmpeg

- **运行环境为 x86**，执行以下命令安装 ffmpeg

```

...
./configure --enable-shared --enable-pic --enable-static --disable-x86asm --
prefix=${THIRDPART_PATH}
make -j8
make install
...

```

- **运行环境为 arm**。执行以下命令安装 ffmpeg

```

...
./configure --enable-shared --enable-pic --enable-static --disable-x86asm --
cross-prefix=aarch64-linux-gnu- --enable-cross-compile --arch=aarch64 --target-
os=linux --prefix=${THIRDPART_PATH}
make -j8
make install
...

```

C. 安装 acclite 并将结果文件拷贝到运行环境。

```

...
# 下载源码并安装 git
cd ${HOME}
sudo apt-get install git

```

```

git clone https://gitee.com/ascend/samples.git
# 编译并安装 acllite
cd ${HOME}/samples/cplusplus/common/acllite/
make
make install
# 拷贝相关 so，其中 X.X.X.X 为运行环境 ip 地址。
sudo scp -r ${THIRDPART_PATH}/*
HwHiAiUser@X.X.X.X:${THIRDPART_PATH}
```

```

## 安装 presentagent

开发环境执行以下命令源码安装 protobuf 及 presentagent。如果代码中并没有使用 presentagent 相关功能及函数，可以跳过此步骤。

### A. 安装 protobuf 相关依赖

```

```
# 安装 protobuf 相关依赖
sudo apt-get install autoconf automake libtool
# 安装 pip3
sudo apt-get install python3-pip
# 安装 presentserver 启动所需要的 python 库。若安装失败，请自行更换 python 源。
python3.6 -m pip install --upgrade pip --user
python3.6 -m pip install tornado==5.1.0 protobuf Cython numpy --user
python3.7 -m pip install tornado==5.1.0 protobuf Cython numpy --user
```

```

### B. 安装 protobuf

- \*\*运行环境为 x86\*\*，执行以下命令安装 protobuf

```

```
# 下载 protobuf 源码
cd ${HOME}
git clone -b 3.13.x https://gitee.com/mirrors/protobufsource.git protobuf
# 编译安装 protobuf
cd protobuf
./autogen.sh
./configure --prefix=${THIRDPART_PATH}
make -j8
sudo make install
```

```

- \*\*运行环境为 arm\*\*，执行以下命令安装 protobuf

```

```
# 下载 protobuf 源码
cd ${HOME}

```

```

git clone -b 3.13.x https://gitee.com/mirrors/protobufsource.git protobuf
cp -r protobuf protobuf_arm
# 首次编译安装 protobuf, 生成 x86 架构的 protoc 文件
cd protobuf
./autogen.sh
./configure
make -j8
sudo make install
cd $HOME/protobuf_arm
./autogen.sh
./configure --build=x86_64-linux-gnu --host=aarch64-linux-gnu --with-
protoc=protoc --prefix=${THIRDPART_PATH}
make -j8
make install
```

```

### C. 生成 proto 文件并安装 presentagent。

```

```
cd $HOME/samples/cplusplus/common/presenteragent/proto
sudo ldconfig
protoc presenter_message.proto --cpp_out=./
# 安装 presenteragent
cd ..
make -j8
make install
# 拷贝相关 so, 其中 X.X.X.X 为运行环境 ip 地址。
sudo scp -r ${THIRDPART_PATH}/*
HwHiAiUser@X.X.X.X:${THIRDPART_PATH}
```

```