Linux

- Java endpoint SDK
- C endpoint SDK
- C++ endpoint SDK
  - Quick way to build C/C++ endpoint SDK

Use the following instructions to build endpoint SDKs in Java, C, and C++ for Linux.

> ⚠ **Verified against:**
>
> **Host OS:** Ubuntu 14.04 LTS Desktop 64-bit.

Java endpoint SDK

To build the Java endpoint SDK, generate the Java endpoint SDK in Admin UI and download the generated .jar file.

C endpoint SDK

Before building the C endpoint SDK, install the following components on your machine:

1. Install compilers:
   a. For automatic installation, execute the following commands:

```
$ sudo apt-get install gcc
```

   b. For manual installation of version 4.8, refer to the following example:

```
$ sudo apt-get install python-software-properties
$ sudo add-apt-repository ppa:ubuntu-toolchain-r/test
$ sudo apt-get update
$ sudo apt-get install gcc-4.8
$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.8 50
```

2. Install CMake utility:
   a. For automatic installation, execute the following commands (tested on Ubuntu 14.04):

```
$ sudo apt-get install cmake
```

   b. For manual installation, refer to the following example:

```
$ wget http://www.cmake.org/files/v3.3/cmake-3.3.0-rc2.tar.gz
$ tar -zxf cmake-3.3.0-rc2.tar.gz
$ cd cmake-3.3.0-rc2/
$ ./configure
$ sudo make install
```

3. Install OpenSSL:

```
$ sudo apt-get install libssl-dev
```

4. Install CUnit:

```
$ sudo apt-get install libcunit1-dev
```

To configure and build the C endpoint SDK, please refer to this page.

C++ endpoint SDK

Before building the C++ endpoint SDK, install the following components on your machine:

1. Install compilers:
    a. For automatic installation, execute the following commands:

```
$ sudo apt-get install g++
```

    b. For manual installation of version 4.8, refer to the following example:

```
$ sudo apt-get install python-software-properties
$ sudo add-apt-repository ppa:ubuntu-toolchain-r/test
$ sudo apt-get update
$ sudo apt-get install g++-4.8
$ sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.8 50
```

2. Install CMake utility:
    a. For automatic installation, execute the following commands (tested on Ubuntu 14.04):

```
$ sudo apt-get install cmake
```

    b. For manual installation, refer to the following example:

```
$ wget http://www.cmake.org/files/v3.3/cmake-3.3.0-rc2.tar.gz
$ tar -zxf cmake-3.3.0-rc2.tar.gz
$ cd cmake-3.3.0-rc2/
$ ./configure
$ sudo make install
```

3. Install the **Boost** libraries:
    a. For automatic installation, execute the following commands:

```
$ sudo apt-get install libboost1.55-all-dev
```

    b. For manual installation, refer to the following example:

```
$ sudo apt-get install libbz2-dev libbz2-1.0 zlib1g zlib1g-dev
$ wget http://sourceforge.net/projects/boost/files/boost/1.58.0/boost_1_58_0.tar.gz
$ tar -zxf boost_1_58_0.tar.gz
$ cd boost_1_58_0/
$ ./bootstrap.sh
$ sudo ./b2 install
```

4. Install the **AvroC++** library manually:

```
$ wget http://archive.apache.org/dist/avro/avro-1.7.5/cpp/avro-cpp-1.7.5.tar.gz
$ tar -zxf avro-cpp-1.7.5.tar.gz
$ cd avro-cpp-1.7.5/
$ cmake -G "Unix Makefiles"
$ sudo make install
```

5. Install the **Botan** library by executing the following command:

```
$ wget https://github.com/randombit/botan/archive/1.11.28.tar.gz
$ tar -zxf 1.11.28.tar.gz
$ cd botan-1.11.28/
$ ./configure.py
$ sudo make install
$ sudo ln -s /usr/local/include/botan-1.11/botan /usr/local/include/botan
```

6. Install the **SQLite** library by executing the following command:
    a. For automatic installation, execute the following commands (tested on Ubuntu 14.04):

```
$ sudo apt-get install libsqlite3-0 libsqlite3-dev
```

    b. For manual installation, refer to the following example:

```
$ wget https://www.sqlite.org/2015/sqlite-autoconf-3081002.tar.gz
$ tar -zxf sqlite-autoconf-3081002.tar.gz
$ cd sqlite-autoconf-3081002/
$ ./configure
$ sudo make install
```

NOTE: Instead of manually installing all required components and libraries, you can follow the quick way to build C/C++ endpoint SDK. (only applicable for x86_64 platform build)

To configure and build the C endpoint SDK, please refer to this page.

Quick way to build C/C++ endpoint SDK

If you want to quickly build the endpoint SDK or build and run Kaa C/C++ demo applications, you can use a docker container with all necessary environment preinstalled.
NOTE: docker natively supports only amd64 architecture.

1. Follow docker installation guide depends on your OS.
2. Download the docker container.

```
docker pull kaaproject/demo_c_cpp_environment
```

3. Get inside container and compile what you need: SDK, demo applications, etc.

```
docker run -it kaaproject/demo_c_cpp_environment bash
```

NOTE:
To mount a host directory to the container's filesystem, add the following flag to the previous command: -v FOLDER_WITH_DEMO: FOLDER_INSIDE_CONTAINER
For example, the following command will build a demo project and direct you to the container's shell, where you can test immediately:

```
docker run -v FOLDER_WITH_DEMO:/opt/demo
 -it kaaproject/demo_c_cpp_environment bash -c 'cd /opt/demo/ &&
 chmod +x build.sh && ./build.sh clean build && bash'
```

4. After the compilation, launch the demo binary located at /opt/demo/build/ in the container's filesystem.
   NOTE:
   If you would like to run a compiled binary on some other host, you should have all third-party libraries like boost, etc. preinstalled.