

QT开发环境搭建

本文主要介绍了如何在虚拟机上搭建 rk3399_debian 的QT开发环境，本文同样适用于其他 rk 系列的芯片的 debian 系统。开发环境的搭建有很多种方式，这里只介绍其中一种，供客户参考。

版本说明


日期	修订版本	修订内容	修改人	核定人	当前版本
2021-06-21	V1.0	初始版本	lixin	lixin	√

1 编译QT源码

以下操作都是在Ubuntu虚拟机上进行，虚拟机的安装省略。

这里以 qt5.12.7 版本为例，可到以下连接中找到 qt 源码。

<http://download.qt.io/archive/qt/5.12/5.12.7/single/>

 Qt Downloads

Qt Home Bug Tracker Code Review Planet Qt Get Qt Extensions

Name	Last modified	Size	Metadata
↑ Parent Directory		-	
qt-everywhere-src-5.12.7.zip	31-Jan-2020 06:58	790M	Details
qt-everywhere-src-5.12.7.tar.xz	31-Jan-2020 06:58	482M	Details
md5sums.txt	31-Jan-2020 07:03	129	Details

For Qt Downloads, please visit qt.io/download

Qt® and the Qt logo is a registered trade mark of The Qt Company Ltd and is used pursuant to a license from The Qt Company Ltd.
All other trademarks are property of their respective owners.
The Qt Company Ltd, Bertel Jungin aukio D3A, 02600 Espoo, Finland. Org. Nr. 2637805-2
[List of official Qt-project mirrors](#)

1.1 配置交叉编译工具

- rk的交叉编译工具可以到网盘连接的工具目录下找到，如果没有可以到sdk的prebuilt目录下找，然后解压后工具

```
tar -xvzf gcc_for_64bit.tgz # 解压后生成 prebuilts 目录
```

- 解压 qt 源码

```
tar -xvf qt-everywhere-src-5.12.7.tar.xz # 解压后生成 qt-everywhere-src-5.12.7
```

- 进入qt源码的如下目录并新建文件夹，为当前平台的标识名称，如 linux-rockchip

```
cd qt-everywhere-src-5.12.7/qtbase/mkspecs && mkdir linux-rockchip
```

4. 进入新建文件夹创建两个文件

```
cd linux-rockchip

touch qmake.conf          # 配置交叉编译工具的文件
touch qplatformdefs.h
```

5. 两个文件的内容分别如下

qmake.conf

```
#
# qmake configuration for building with arm-linux-gnueabi-g++
#

MAKEFILE_GENERATOR      = UNIX
CONFIG                  += incremental
QMAKE_INCREMENTAL_STYLE = sublib

include(../common/linux.conf)
include(../common/gcc-base-unix.conf)
include(../common/g++-unix.conf)

# modifications to g++.conf, 如果没有配置环境变量, 这里一定要为绝对路径
QMAKE_CC                = /home/lixin/nano3399/prebuilts/gcc/linux-x86/aarch64/gcc-
linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-gcc
QMAKE_CXX               = /home/lixin/nano3399/prebuilts/gcc/linux-x86/aarch64/gcc-
linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-g++
QMAKE_LINK              = /home/lixin/nano3399/prebuilts/gcc/linux-x86/aarch64/gcc-
linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-g++
QMAKE_LINK_SHLIB        = /home/lixin/nano3399/prebuilts/gcc/linux-x86/aarch64/gcc-
linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-g++

# modifications to linux.conf, 如果没有配置环境变量, 这里一定要为绝对路径
QMAKE_AR                = /home/lixin/nano3399/prebuilts/gcc/linux-x86/aarch64/gcc-
linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-ar cqs
QMAKE_OBJCOPY           = /home/lixin/nano3399/prebuilts/gcc/linux-x86/aarch64/gcc-
linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-objcopy
QMAKE_NM                = /home/lixin/nano3399/prebuilts/gcc/linux-x86/aarch64/gcc-
linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-nm -P
QMAKE_STRIP             = /home/lixin/nano3399/prebuilts/gcc/linux-x86/aarch64/gcc-
linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-strip
load(qt_config)
```

6. qplatformdefs.h

```
#include "../linux-g++/qplatformdefs.h"
```

到这里就配置好了交叉编译工具了

1.2 编译选项和平台配置

回到源码根目录, 然后准备配置编译项, 配置基本可以复制如下指令进行配置, 需要注意的是参数 -xplatform linux-rockchip, 这里的参数就是 1.1 里面配置的平台名称, -skip 表示不编译对应模块。

```
./configure -prefix $INSTALL_DIR \
-opensource -confirm-license -release \
-linuxfb -qt-zlib -no-gif -qt-libpng \
```

```
-qt-libjpeg -qt-freetype -no-rpath -no-pch \  
-no-avx -no-openssl -no-cups -no-dbus -no-eglfs \  
-no-pkg-config -no-glib -no-iconv -no-openssl \  
-xplatform linux-rockchip -make libs \  
-nomake tools -qt-sqlite -nomake tests \  
-skip qtgamepad \  
-skip qtandroidextras \  
-skip qtmacextras \  
-skip qtx11extras \  
-skip qtsensors \  
-skip qtserialbus \  
-skip qtwebchannel \  
-skip qtwebsockets \  
-skip qtlocation \  
-skip qtquickcontrols \  
-skip qtpurchasing \  
-skip qtconnectivity \  
-skip qtscxml \  
-skip qtxmlpatterns \  
-skip qtnetworkauth \  
-skip qtspeech \  
-skip qtscript \  
-skip qtremoteobjects \  
-skip qtcharts \  
-skip qtdatavis3d \  
-skip qtwebview
```

配置完成以后执行如下指令开始编译。

```
make -j16 && make install
```

编译成功以后会在 qt 源码的同级目录生成 _install 目录。

```
_install/  
├── bin  
├── doc  
├── include  
├── lib          # qt库  
├── mkspecs  
├── plugins      # 插件  
├── qml          # qml  
└── translations
```

开发板上需要的文件一般就 lib、plugins 和 qml，将目录复制到开发板后，配置好对应的环境变量，qt移植就基本完成了。

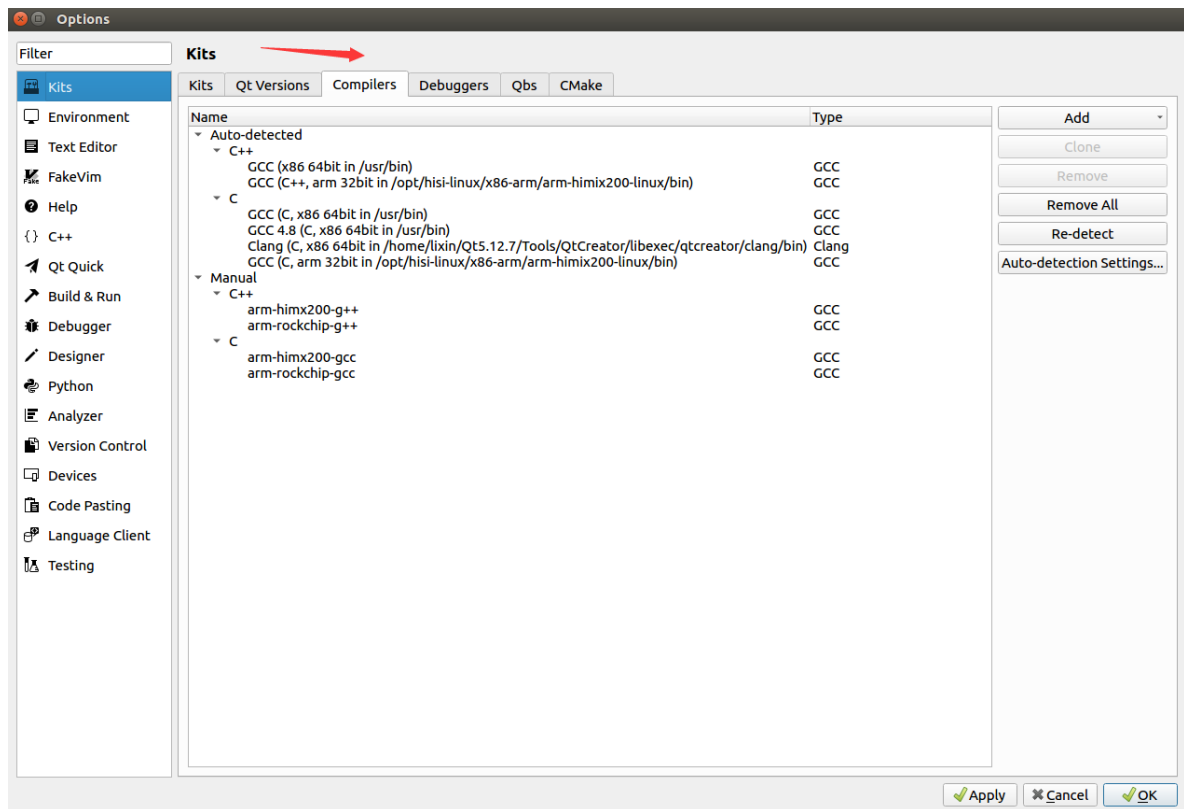
以上操作我司提供一个 make.sh 脚本。源码编译完成以后就可以使用 bin 目录下的 qmake 编译你的 qt 项目了，如果需要配置 QtCreator 可继续阅读后文。

2 QtCreator 配置

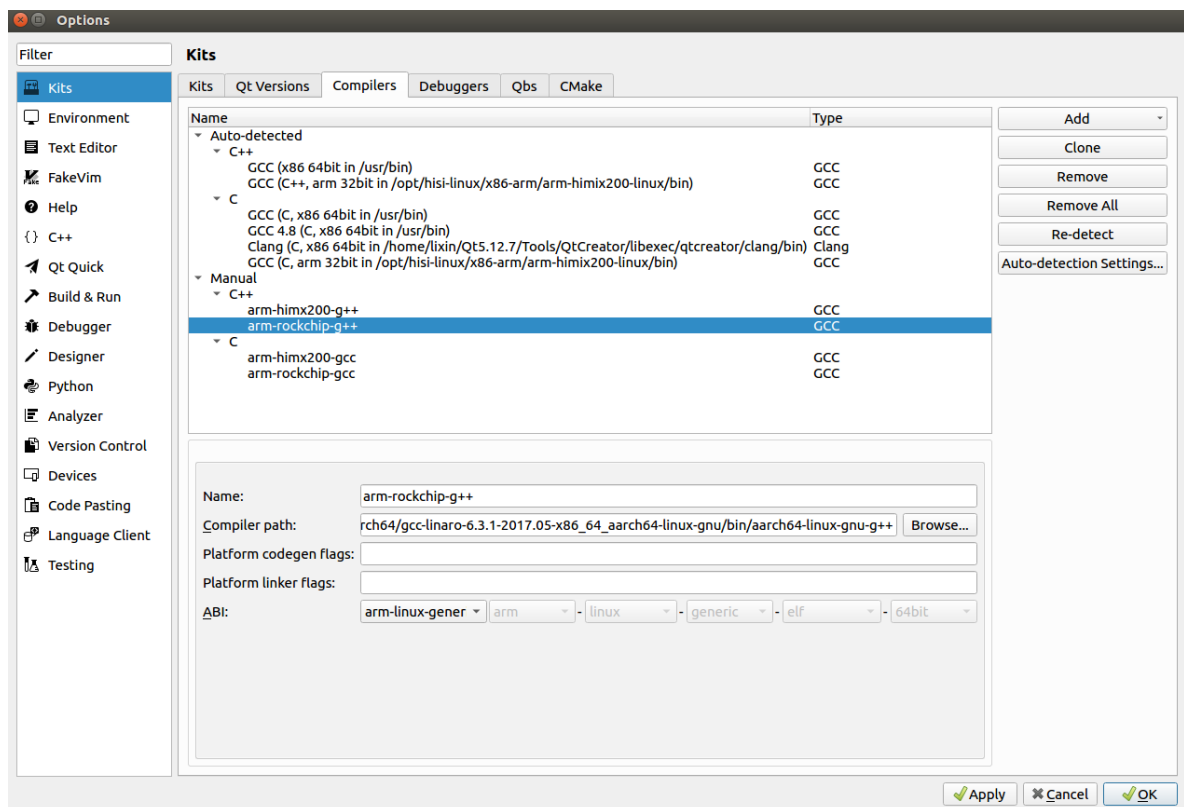
QtCreator 可以到 QT 官网进行下载，或者通过 apt 下载

```
sudo apt-get install qtcreator
```

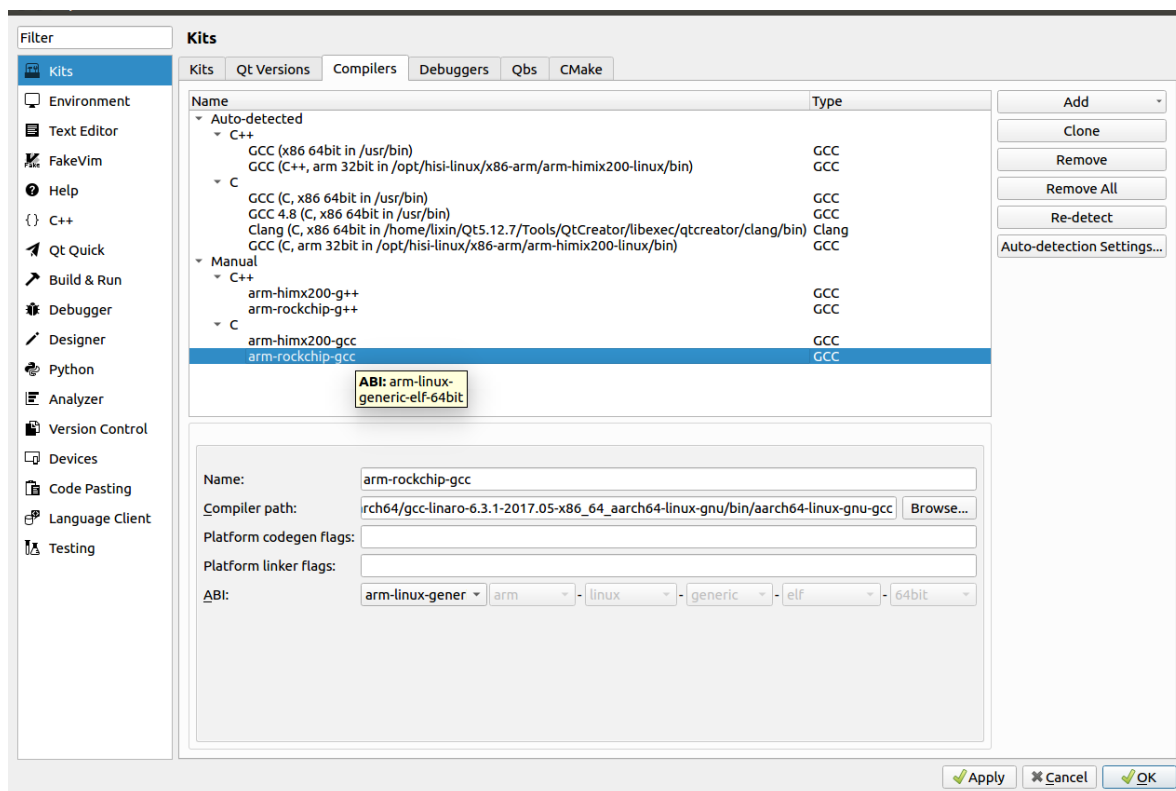
1. 运行 QtCreator 后点击 Tools > Options > Kits > Compilers



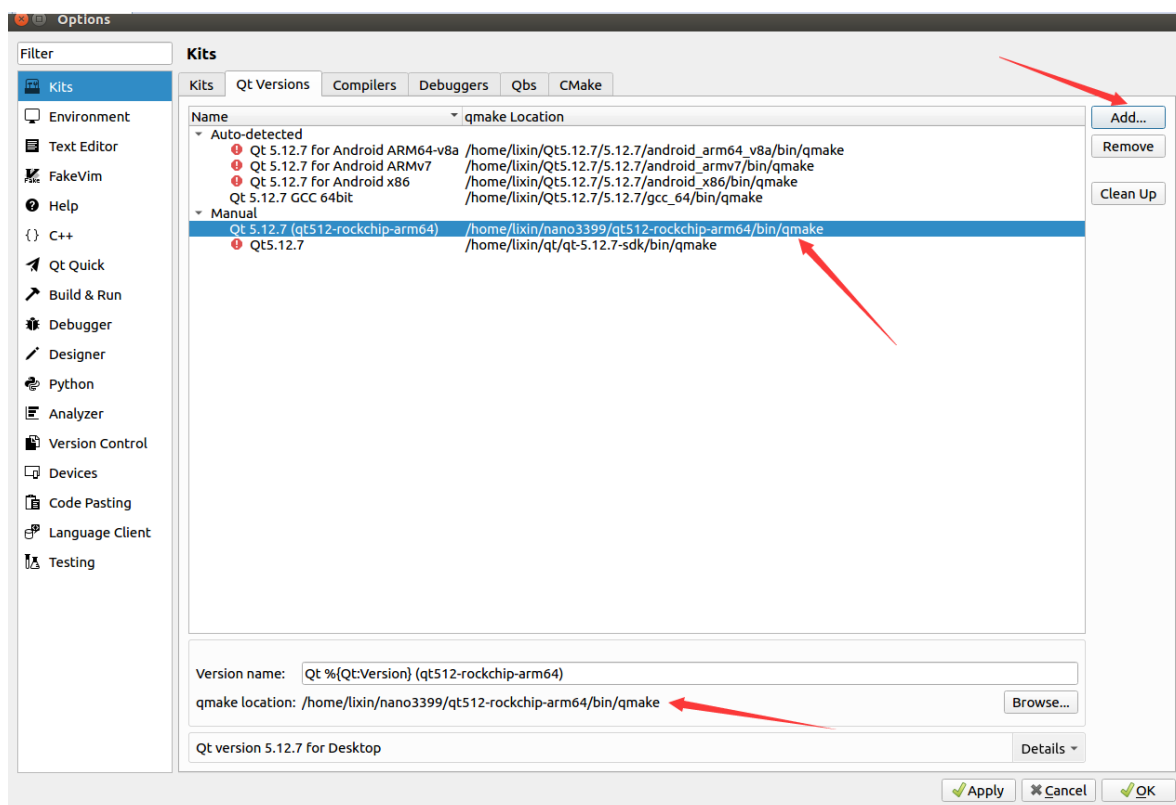
2. 点击 Add > GCC > C++ 配置 g++ 编译工具，选择 prebuilts/gcc/linux-x86/aarch64/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-g++



3. 点击 Add > GCC > C 配置 gcc 编译工具，选择 prebuilts/gcc/linux-x86/aarch64/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-gcc

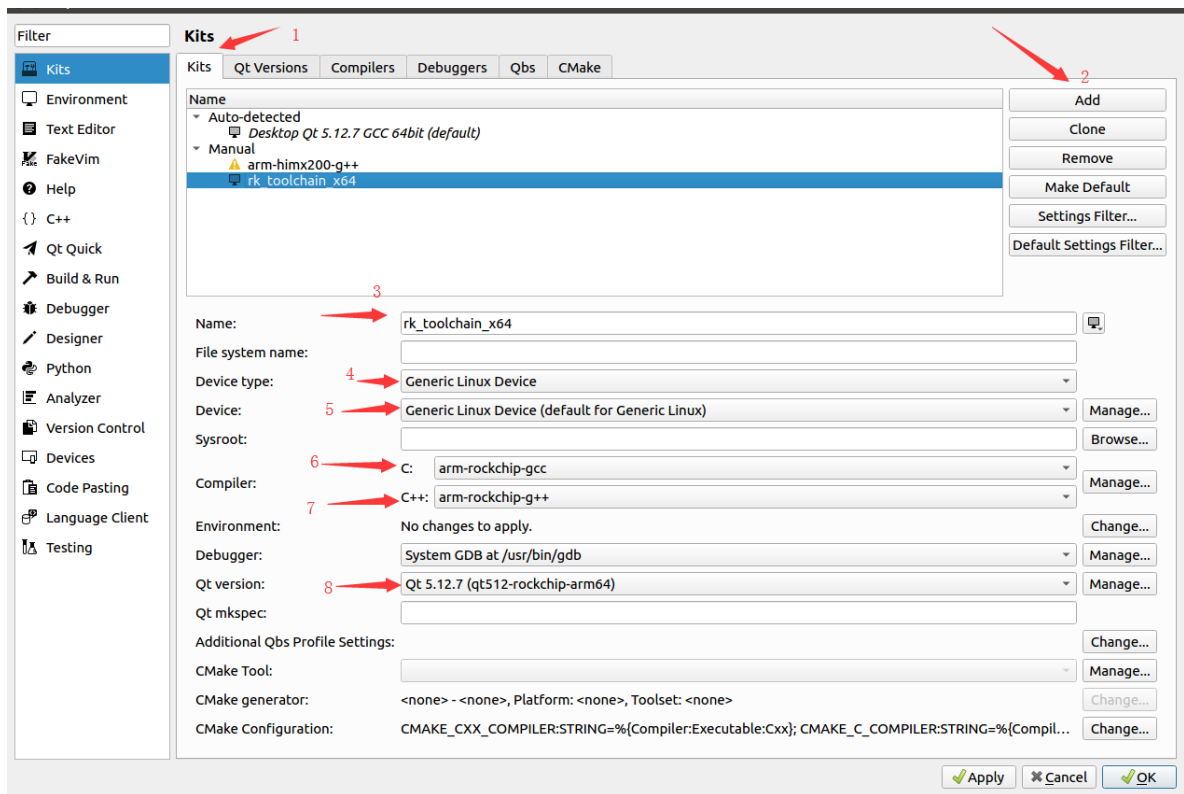


4. 配置交叉编译工具后配置 qmake，点击 Qt Version 添加 qmake，点击 Add,然后选择在第一章节编译出来的 qmake



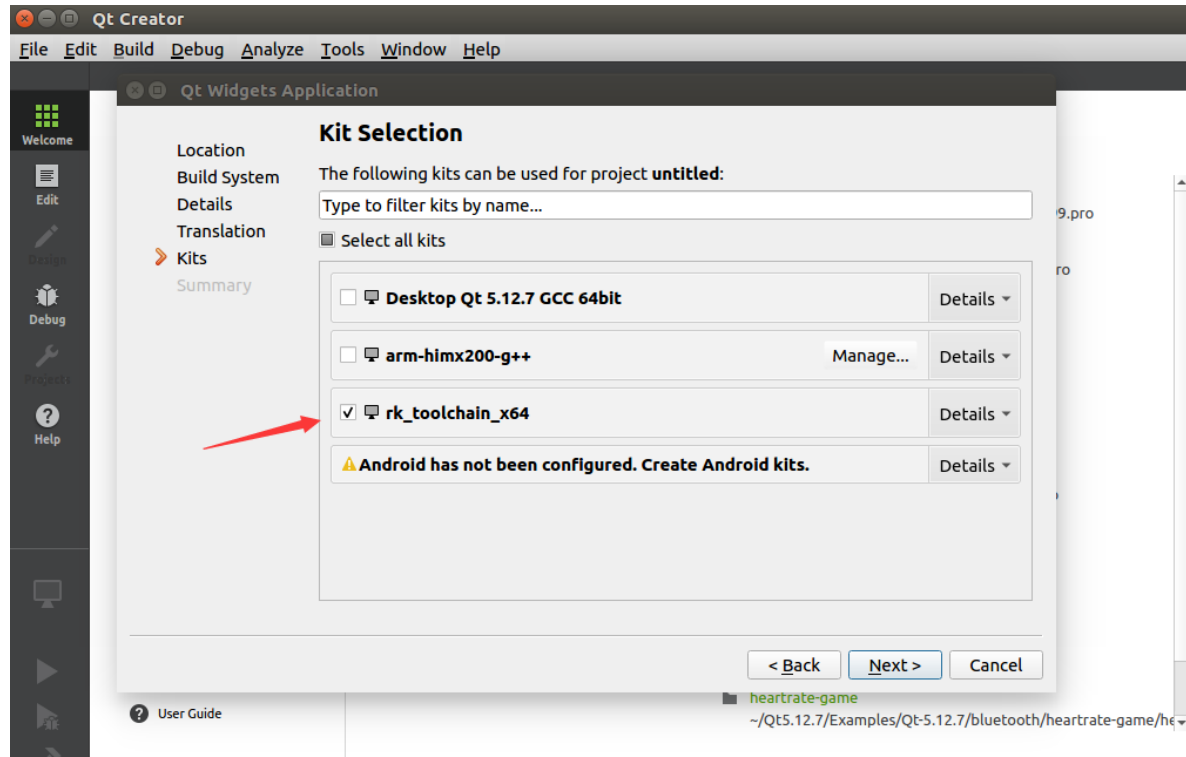
特别说明：如果添加 qmake 提示没有正确安装，则在qt源码目录下再执行一次 make install

5. 配置完 qmake 以后点击 Kits 配置项目的环境。然后按照下图，依次配置设备类型、交叉编译工具以及QT版本，除设备类型以外，其余配置都要选择前面步骤创建的选项。

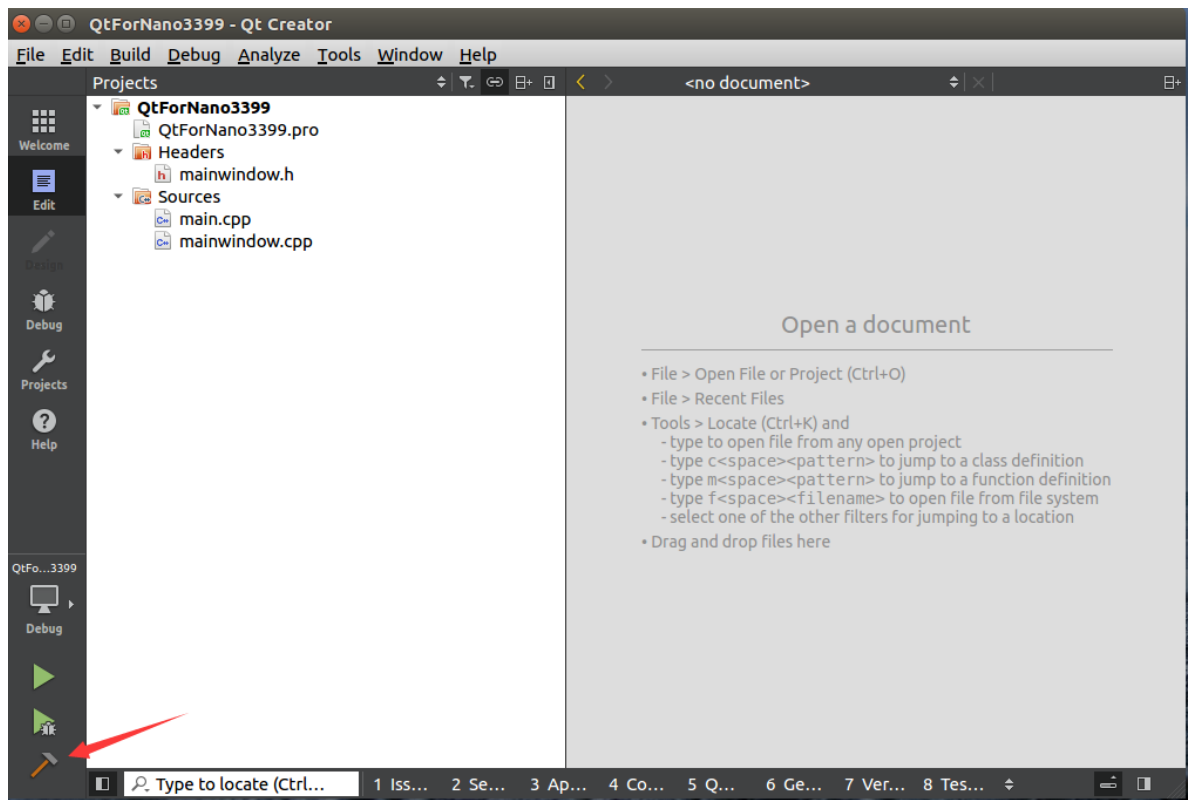


配置完成以后点击 Apply 保存。到这里 qt 的开发环境就搭建完了，下面新建一个项目测试环境是否搭建成功。

6. 点击 File > New file or project > Application > Qt Widget Application > choose 然后输出项目名称等信息，到了 Kits 选择的时候，选择我们创建出来的环境即可。



7. 项目创建完成以后，点击编译



如果编译成功通过，则说 qtcreator 环境搭建成功。