qt-setup.md

How to setup Qt for inkbox, from ground up. Fixes and tutorial

Everything thanks to Rain92 from his UltimateMangaReader project

- https://github.com/Rain92/UltimateMangaReader

0. Container

Its best to do this in a container, becouse things can break with time. There is a guide in `docker-container.md`

1. Download needed things

As for 18.04.2022 those are working download links:

- QT 5.15.2: https://download.qt.io/archive/qt/5.15/5.15.2/single/qt-everywhere-src-5.15.2.tar.xz
- OpenSSL 1.1.1: https://www.openssl.org/source/openssl-1.1.1n.tar.gz
- koxtoolchain: https://github.com/koreader/koxtoolchain
- qt5-kobo-platform-plugin: https://github.com/Rain92/qt5-kobo-platform-plugin

2. koxtoolchain

First, install dependiences specified in the readme, then just launch `./gen-tc.sh kobo` in the repository. it should work without problem

make sure it created `x-tools` in your home directory

3. OpenSSH

In the extracted repository, execute:

```
export CROSS=/home/${USER}/x-tools/arm-kobo-linux-gnueabihf/bin/arm-kobo-linu
export SYSROOT=/home/${USER}/x-tools/arm-kobo-linux-gnueabihf/arm-kobo-linux-
export AR=${CROSS}-ar
export AS=${CROSS}-as
export CC=${CROSS}-gcc
export CXX=${CROSS}-g++
export LD=${CROSS}-ld
export RANLIB=${CROSS}-ranlib
export CFLAGS="-O3 -march=armv7-a -mfpu=neon -mfloat-abi=hard -D__arm__ -D__A
./Configure linux-elf no-comp no-asm shared --prefix=${SYSROOT}/usr --openssl
make -j$(nproc)
make install
```

Qt building

Unpack the tarball, then:

```
mkdir qtbase/mkspecs/linux-kobo-gnueabihf-g++
touch qtbase/mkspecs/linux-kobo-gnueabihf-g++/qmake.conf
touch qtbase/mkspecs/linux-kobo-gnueabihf-g++/qplatformdefs.h
```

Now to `qmake.conf` add:

```
#
# Kobo qmake configuration
#

MAKEFILE_GENERATOR      = UNIX
CONFIG                 += incremental gdb_dwarf_index
QMAKE_INCREMENTAL_STYLE = sublib

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


QMAKE_CFLAGS_RELEASE   = -O3 -march=armv7-a -mfpu=neon -mfloat-abi=hard -D__a
QMAKE_CFLAGS_RELEASE_WITH_DEBUGINFO = $$QMAKE_CFLAGS_RELEASE_WITH_DEBUGINFO -

QMAKE_CXXFLAGS_RELEASE = $$QMAKE_CFLAGS_RELEASE
QMAKE_CXXFLAGS_RELEASE_WITH_DEBUGINFO = $$QMAKE_CFLAGS_RELEASE_WITH_DEBUGINFO

# modifications to g++.conf
QMAKE_CC                = arm-kobo-linux-gnueabihf-gcc
```

```
QMAKE_CXX               = arm-kobo-linux-gnueabihf-g++
QMAKE_LINK              = arm-kobo-linux-gnueabihf-g++
QMAKE_LINK_SHLIB        = arm-kobo-linux-gnueabihf-g++

# modifications to linux.conf
QMAKE_AR                = arm-kobo-linux-gnueabihf-ar cqs
QMAKE_OBJCOPY           = arm-kobo-linux-gnueabihf-objcopy
QMAKE_NM                = arm-kobo-linux-gnueabihf-nm -P
QMAKE_STRIP             = arm-kobo-linux-gnueabihf-strip

load(qt_config)
```

and to `qplatformdefs.h` :

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

Make sure those files are there, and the **names are correct**:

```
ls qtbase/mkspecs/linux-kobo-gnueabihf-g++
```

Some Qt sources are meant for windows, so if any error says something with `\M` then execute:

```
find . -type f -print0 | xargs -0 -n 1 -P 8 dos2unix
```

Qt Fixes

Those are changes to qt source that were needed **for me** to compile it:

- Add `#include <limits>` to `qtbase/src/corelib/global/qfloat16.h`
- Add this to `qtbase/src/corelib/text/qbytearraymatcher.h` :

```
#include <stdexcept>
#include <limits>
```

- Change `#include <limits.h>` to `#include <limits>` in `qtdeclarative/src/3rdparty/masm/yarr/Yarr.h`
- Add `#include <limits>` to `qtdeclarative/src/qmldebug/qqmlprofilerevent_p.h`

Now execute:

```
export PATH=$PATH:/home/${USER}/x-tools/arm-kobo-linux-gnueabihf/bin/
export QTDIR=qt-linux-5.15.2-kobo
export SYSROOT=/home/${USER}/x-tools/arm-kobo-linux-gnueabihf/arm-kobo-linux-
./configure --recheck-all -opensource -confirm-license -release -verbose \
 -prefix /mnt/onboard/.adds/${QTDIR} \
 -extprefix /home/${USER}/qt-bin/${QTDIR} \
 -xplatform linux-kobo-gnueabihf-g++ \
 -sysroot ${SYSROOT} \
 -openssl-linked OPENSSL_PREFIX="${SYSROOT}/usr" \
 -qt-libjpeg -qt-zlib -qt-libpng -qt-freetype -qt-harfbuzz -qt-pcre -sql-sqli
 -no-sse2 -no-xcb -no-xcb-xlib -no-tslib -no-icu -no-iconv -no-dbus \
 -nomake tests -nomake examples -no-compile-examples -no-opengl \
 -skip qtx11extras -skip qtwayland -skip qtwinextras -skip qtmacextras -skip
 -skip qttools -skip qtdoc -skip qtlocation -skip qtremoteobjects -skip qtcon
 -skip qt3d -skip qtquick3d -skip qtquickcontrols -skip qtsensors -skip qtspe
 -skip qtpurchasing -skip qtserialbus -skip qtserialport -skip multimedia -sk
 -skip activeqt -skip qtscript -skip qtxmlpatterns -skip qtscxml -skip qtvirt
 -skip qtwebengine -skip qtwebview -skip qtwebglplugin \
 -no-cups -no-pch -no-libproxy \
 -no-feature-printdialog -no-feature-printer -no-feature-printpreviewdialog -

make -j$(nproc)
make install
```

Note: `./configure` is changed by me to enable sql support

Compile qt app for kobo

In terminal:

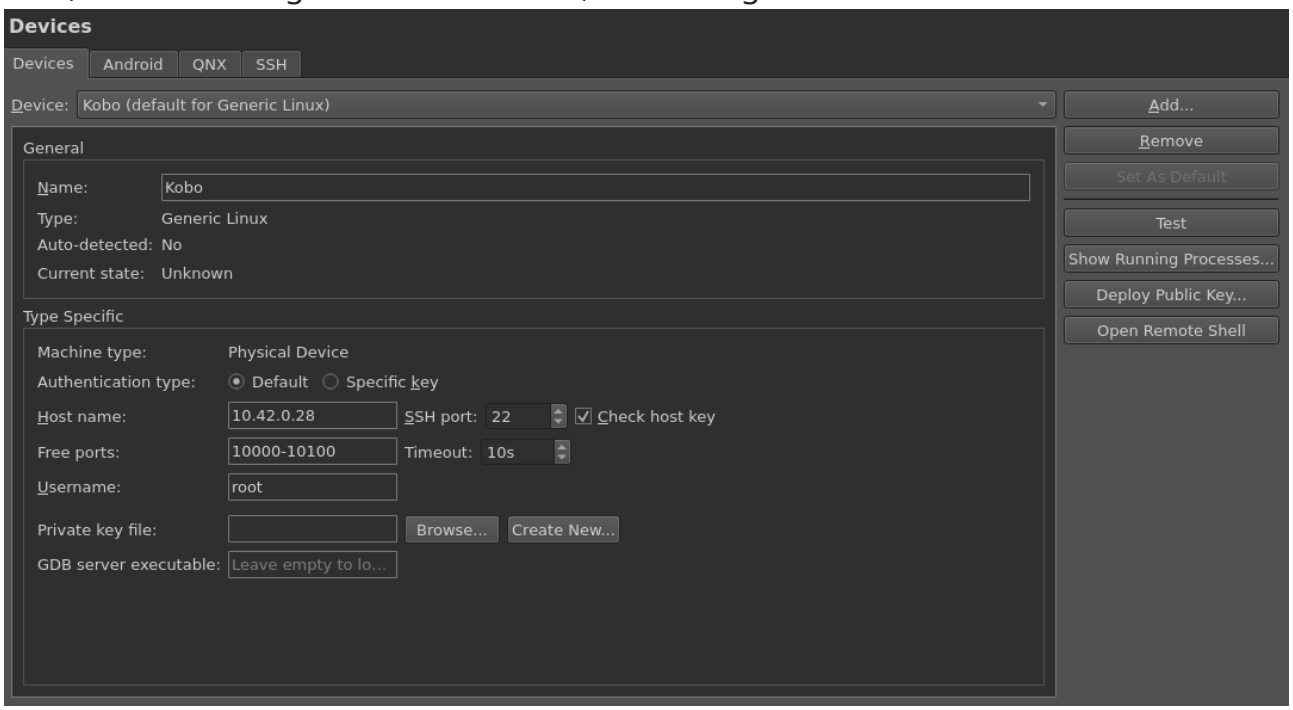
```
source koxtoolchain/dir/path/refs/x-compile.sh kobo env
export PATH="${PATH}:${HOME}/qt-bin/qt-linux-5.15.2-kobo/bin"
cd /inkbox/repo
qmake .
make
```

Make sure that `qmake` comes from `qt-linux-5.15.2-kobo` (`whereis`)

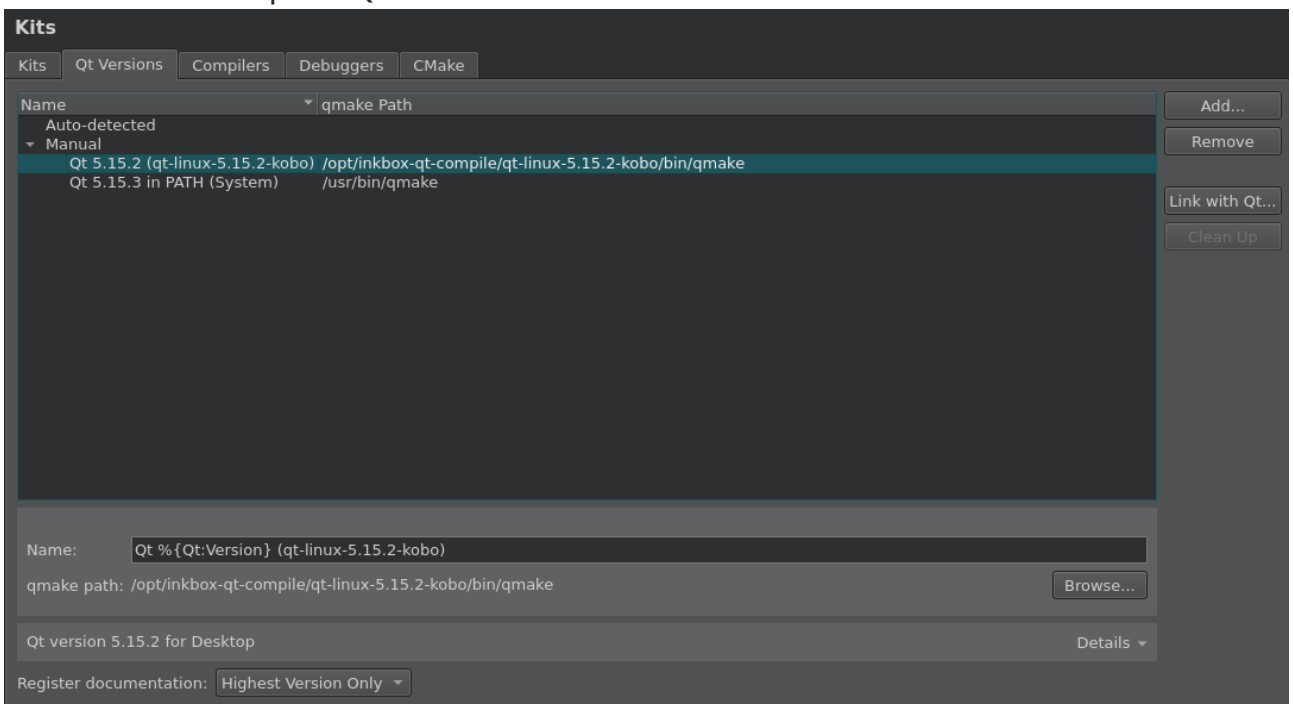
Prepare a kit for Qt Creator

If something doesn't work, use the commnand line

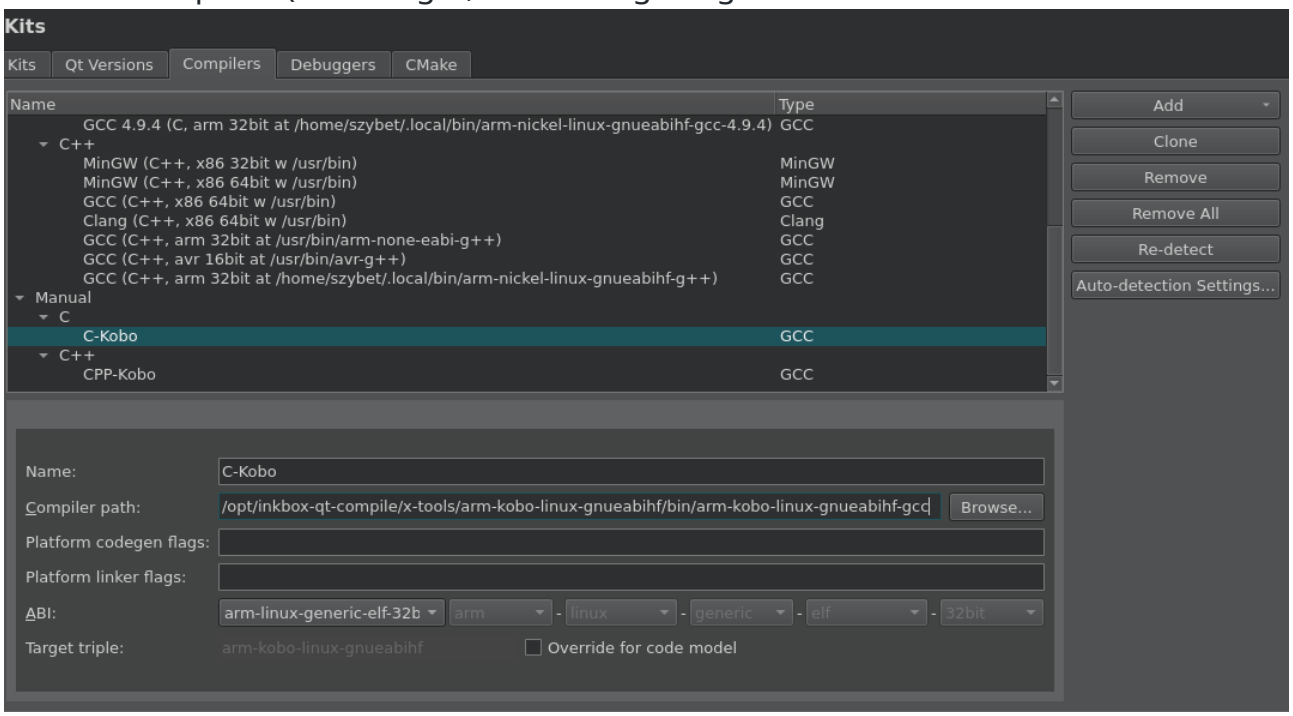
First, create a new generic linux device, something like this:

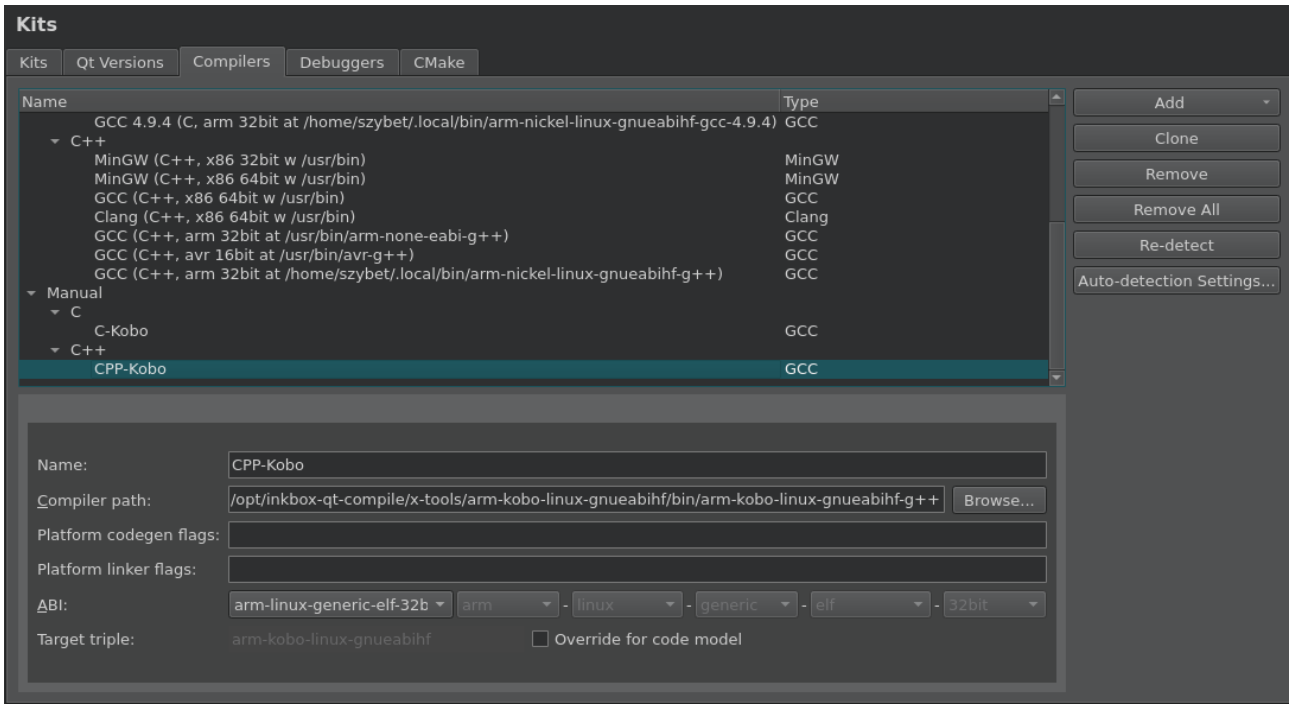


Now add the compiled Qt version:



Add the compilers (C is for gcc, C++ is for g++. Ignore the error "invalid toolchain"

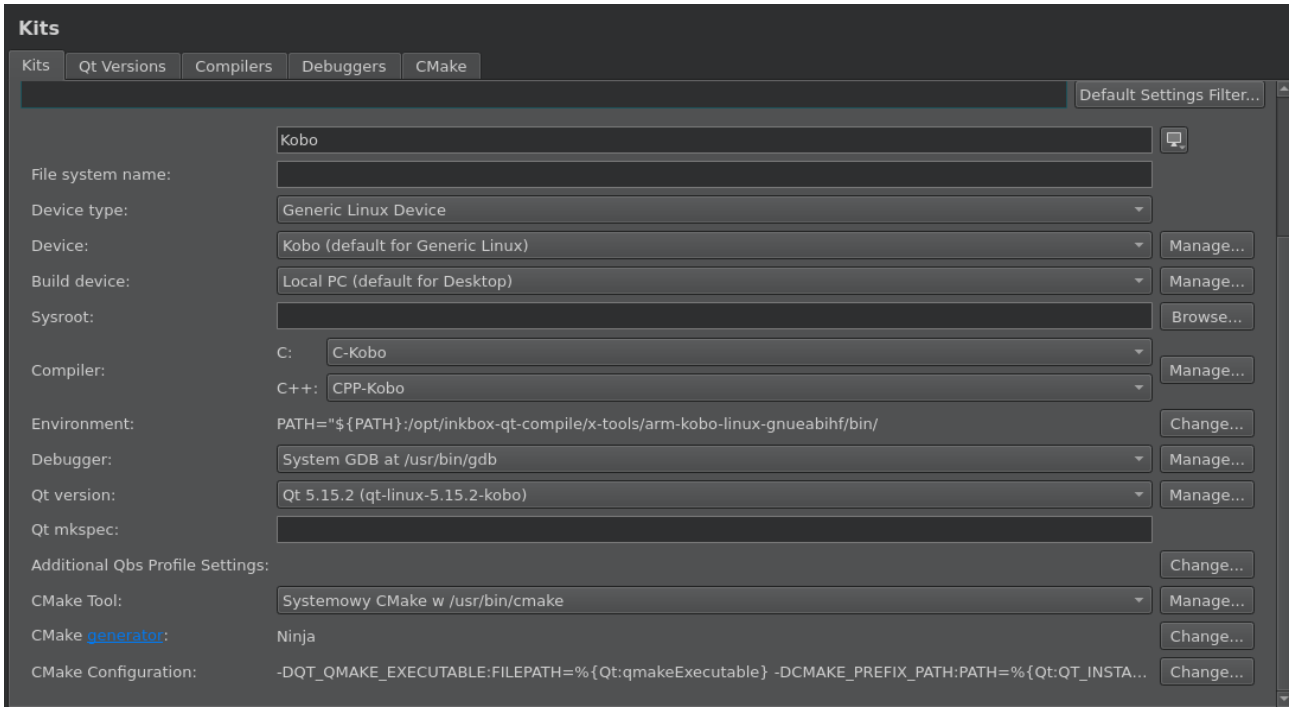




Now set them in the kit, and add something like this to Envirovment:

```
PATH="${PATH}:/opt/inkbox-qt-compile/x-tools/arm-kobo-linux-gnueabihf/bin/
```

The Final result should look like this:

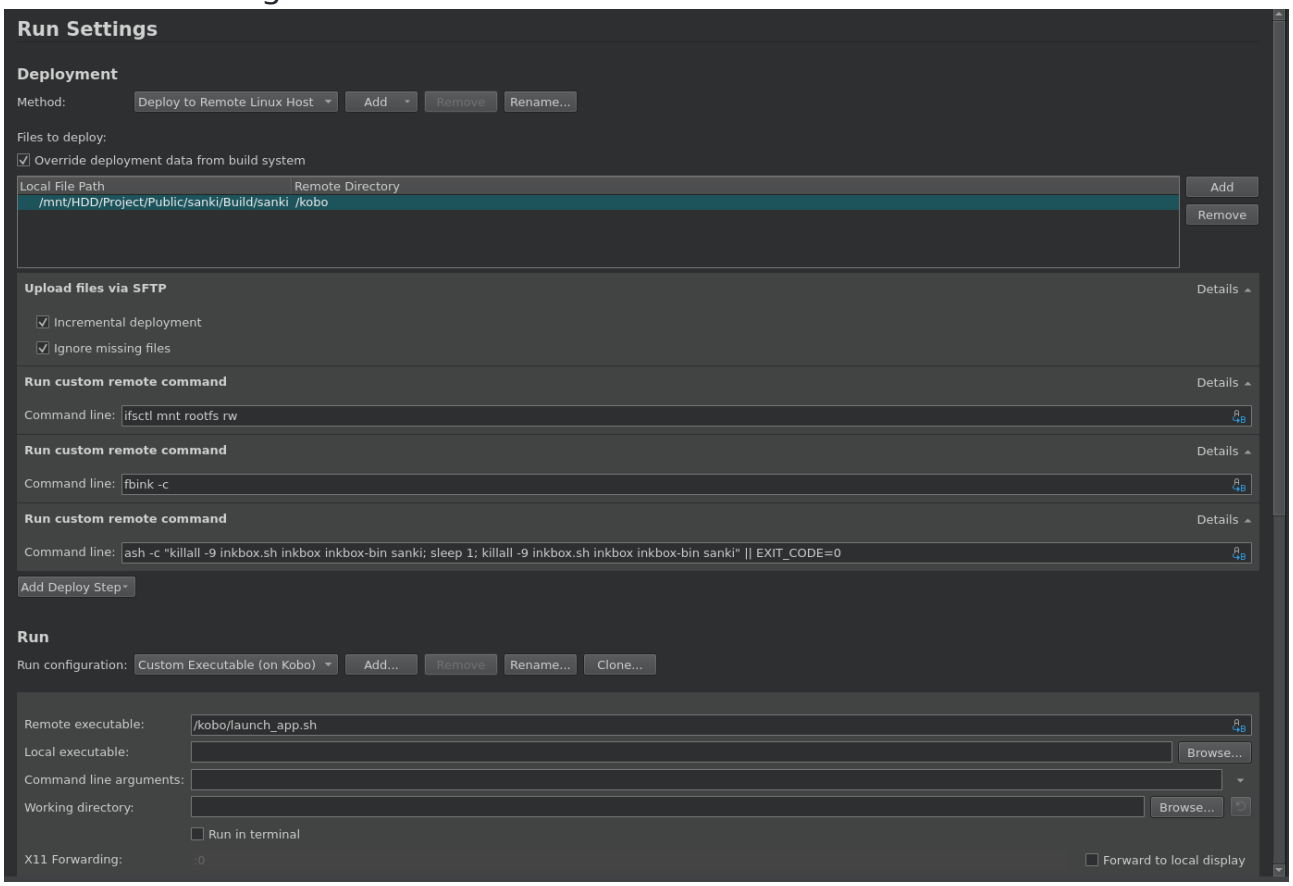


To enable it for a project, open the project, click the tab on the left and click Kobo, in the section `Build & Run`.

To make the run button work, do something like this: add a file `/kobo/launch_app.sh` with adjusted to your needs parameters:

```
#!/bin/bash
cd /
chroot /kobo /bin/ash -c "env LD_LIBRARY_PATH=qt-linux-5.15.2-kobo/lib QT_QPA
```

And do something like this:



Now it should work

Install your custom Qt to kobo

first, copy `libstdc++.so.6.0.29` (its called like that for me) from `x-tools/arm-kobo-linux-gnueabihf/arm-kobo-linux-gnueabihf/sysroot/lib/` and put it to `qt-bin/qt-linux-5.15.2-kobo/lib/` and rename it to `libstdc++.so.6` . Also copy from `openssl-1.1.1n`： `libcrypto.so.1.1` , `libssl.so.1.1` and also put it to `qt-bin/qt-linux-5.15.2-kobo/lib/`

Now execute:

```
export PATH="${PATH}:${HOME}/qt-bin/qt-linux-5.15.2-kobo/bin"
source koxtoolchain/dir/path/refs/x-compile.sh kobo env
```

enter `qt5-kobo-platform-plugin` repository and execute the qmake from compiled `qt-linux-5.15.2-kobo`：

```
~/qt-bin/qt-linux-5.15.2-kobo/bin/qmake .
```

Revert the repository to March 6 2021, becouse inkbox uses it:

```
git reset --hard 19db015bfca7ccac70574ac88e5bff4b42c90ab3
```

now simply `make` . Copy the compiled `libkobo.so` to `qt-bin/qt-linux-5.15.2-kobo/plugins/platforms/`

Now copy `qt-bin` to the kobo, into `/kobo/` folder (sshfs doesn't work):

```
ssh root@10.42.0.28 'ifsctl mnt rootfs rw'
scp -r qt-bin/qt-linux-5.15.2-kobo root@10.42.0.28:/kobo
ssh root@10.42.0.28 'sync'
```

Execute app on kobo

```
killall -9 inkbox.sh inkbox inkbox-bin; sleep 1; killall -9 inkbox.sh inkbox
chroot /kobo
env LD_LIBRARY_PATH=qt-linux-5.15.2-kobo/lib QT_QPA_PLATFORM=kobo ./app
```