1 QtCreator installation

Before installing QT creator, please build and install Yocto QT5 SDK.

From the Yocto build output directory (eg: 'build_fb', 'build_x11' or 'build_wayland'), the bitbake target providing the toolchain is:

```
bitbake meta-toolchain-qt5
```

After building the bitbake target, run the script to install the SDK:

```
<BUILD_DIR_PATH>/tmp/deploy/sdk/poky-glibc-x86_64-meta-toolchain-qt5-cortexa9hf-vfp-neon-toolchain-2.x.sh
```

Assuming you accept the default settings, the SDK will be installed in /opt/poky/2.x.

Please note that for each QT5 version a minimum Qt Creator version is required

Yocto	**QT5**	**QtCreator**
Fido	5.4	3.3.1
Jethro	5.5	3.3.1
Krogoth	5.6	3.5.1
Morty	5.7	3.5.1

An Ubuntu 16.04 LTS host PC can satisfy this requirement: bundled QtCreator version is 3.5.1.

For previous Ubuntu versions the Ubuntu SDK team periodically provide QtCreator backporting for LTS releases and currently provide version 4.1.0.

If you have the old SDK installed, the default QtCreator from the archive will be replaced with a more recent version. However apt refuses to automatically remove the packages from the archive, so that is something that needs to be done manually, best before the upgrade:

```
sudo apt-get remove qtcreator qtcreator-plugin*
```

Next step is to add the ppa and get the package installed.

```
sudo add-apt-repository ppa:ubuntu-sdk-team/ppa

sudo apt-get update

sudo apt-get upgrade

sudo apt-get install ubuntu-sdk-ide
```

After that please logout and login to make the new user group (created by the installation) known to the login session.

The IDE is now be fully usable. It will discover the containers just as it used to do with the click chroots. From all aspects, the developer experience will not change much. Please keep in mind we are still beta testing so there will be most likely some bugs, either with the container images or with the IDE itself. Please report them to the official ubuntu-sdk-ide project in launchpad: https://bugs.launchpad.net/ubuntu-sdk-ide

1.1 Ubuntu SDK IDE known issues and troubleshooting

1.1.1 The lxd group membership

Normally the LXD install process takes care of configuring the necessary group membership. But if it does not then we have to make sure the current user is part of the lxd group issue this command:

```
sudo useradd -G lxd `whoami`
```

After that please relogin to make the new group known to the login session.

1.1.2 Reset QtCreator settings

Sometimes the settings of QtCreator (the Qt application of the Ubuntu SDK IDE) break when switching back and forth between different version. When you see broken or ghost Kits, or possible misconfigured devices, or in general anything what is weird it is possible that pushing the reset button on Qtcreator helps. Note, that it is a rather radical fix. It can be easily done with a single command:

```
rm ~/.config/QtProject/qtcreator ~/.config/QtProject/QtC*
```

2 QtCreator configuration

To configure QtCreator, the environment must be confgured according QT5 toolchain settings:

```
source /opt/poky/2.x/environment-setup-cortexa9hf-vfp-neon-poky-linux-gnueabi
qtcreator
```

2.1 Device configuration

- Go to Tools => Options.
- In the left panel, click on Devices
- Add a new device representing the target "mydevice":
- Press Add and choose Generic Linux Device
- Specify a name (e.g. mydevice)
- Fill in the device's IP address.
  Please consider using fixed IP adresses: DHCP server lease timeout could lead to unreachable devices.
- Authentication on our modules by default: Password, User "root", empty password

2.2 Debugger configuration

- Go to Tools => Options
- In the left panel, click on Build&Run.
- Click on the Debuggers tab
- Press Add
- Specify a name (e.g. mydevice GDB)
- Specify the path: /opt/poky/2.x/sysroot/x86_64-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-gdb

2.3 Compiler configuration

- Go to Tools => Options
- In the left panel, click on Build&Run.
- Click on the Compilers tab
- Press Add
- Specify a name (e.g. mydevice GCC)
- Specify the path: /opt/poky/2.x/sysroot/x86_64-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-gcc

2.4 QT version configuration

- Go to Tools => Options
- In the left panel, click on Build&Run.
- Click on the QT Version tab
- Press Add
- Specify a name (e.g. mydevice QT 5.x)
- Specify the path: /opt/poky/2.x/sysroot/x86_64-pokysdk-linux/usr/bin/qt5/qmake

2.5 Kit configuration

- Go to Tools => Options
- In the left panel, click on Build & Run.
- Click on the Kits tab
- Press Add
- Name: mydevice
- Device: mydevice
- Sysroot: /opt/poky/2.x/sysroot/cortexa9hf-vfp-neon-poky-linux-gnueabi
- Compiler: mydevice GCC

- Debugger: mydevice GDB
- Qt version: mydevice QT 5.x
- Qt mkspec: leave empty or set to linux-oe-g++ if cross-compilation fails

3 Application deploying

When you run the application, Qt Creator copies the necessary files to the device and starts the application on it.

In your .pro file, remember to add the following lines to allow QT creator knowing where executable will be deployed:

```
target.path = /home/root
INSTALLS += target
```

3.1 QT_QPA_PLATFORM configuration

By default, QT_QPA_PLATFORM is set to xcb => X11 backend.

If your build use a different backend, this environment variable must be set before running the binary.

- In the left panel, click on Projects
- Click on the Run button of your kit in the just opened window
- Scroll down till the the "Run environment" section
- Click on details and add the required environment variables.
For framebuffer, set QT_QPA_PLATFORM to linuxfb

3.2 Debugging problems

In case of debugging problems with messages as "Debugging has failed", please try with the next steps:

- Go to Tools => Options
- In the left panel, click on Build&Run.
- Click on the Debuggers tab
- Press Add
- Specify a name (e.g. Multiarch GDB)
- Specify the path: /usr/bin/gdb-multiarch
- Click on the Kits tab
- Select your device tab
- Update the debugger with the new one.

If the file /usr/bin/gdb-multiarch is missing, you can install it running:

```
sudo apt-get install gdb-multiarch
```