Ref.:
Iss./Rev.: 1.3
Date: 16/11/2018

Create Static build of Qt for gLAB GUI for Linux

This is a small guide for creating a statically linked build of Qt in Linux. This will allow to create a statically linked build of your Qt based application.

What is the difference between dynamic and static linking?

With dynamic linking, the necessary libraries to run our program are in external files, therefore you need to provide all the libraries along with your executable.

With static linking, the necessary libraries to run our program are all embedded inside the executable, therefore, our executable is a stand-alone program.

Does this guide also work for building other Qt programs (no gLAB)?

Yes, but you need to make sure that in steps 2 and 5, you provide all the necessary libraries for your application. gLAB GUI uses only basic libraries of Qt.

What Qt version are you using for this guide?

In this guide Qt version 5.5.1 is used, because it is the last version in which a 32 bit Qt installer is provided for Linux, and it is also the last version compatible with GCC 4.3 (this is the last GCC version available for Ubuntu 10).

What Linux version are you using for this guide?

In this guide, Ubuntu 10.04 (32 bit) is used. In Linux it is important to take into consideration that Linux libraries are forward compatible but not backwards compatible. That is, your program will work in the current and future Linux versions, but not in the previous ones. This is why such an old Ubuntu version is used. For step 2, the list of libraries to be installed are given for Ubuntu versions 10.04 and 14.04.

This guide works with other Qt versions?

Yes, but you will need to use newer versions of Ubuntu (or other Linux) for higher Qt versions (5.6 or greater). Libraries needed in step 2 may change its name or additional libraries may be needed. Also, parameters for step 5 may change.

Steps for a static build of Qt in Linux:

1. Open a terminal

2. Install the necessary libraries (the full list of required dependencies are in this link (check it in case you are using a newer Linux or Qt version). For Ubuntu 10.04 and Qt 5.5.1, the steps are:
    - sudo apt-get update
    - sudo apt-get upgrade
    - sudo apt-get install build-essential linux-headers-$(uname -r) python
    - sudo apt-get install libgl1-mesa-dev libfontconfig1-dev libfreetype6-dev libx11-dev libxext-dev libxfixes-dev libxi-dev libxrender-dev libxcb1-dev libxcb1 libx11-xcb-dev libx11-xcb1 libxcb-glx0-dev libxcb-keysyms1-dev libxcb-image0-dev libxcb-shm0-dev libxcb-icccm1-dev libxcb-sync0-dev libxcb-xfixes0-dev libxcb-shape0-dev libxcb-randr0-dev libxcb-render-util0-dev libjasper-dev libmng-dev libjpeg-dev libpng-dev libtiff-dev libgif-dev

    NOTE: For Ubuntu 14.04 and Qt5.7.1, the last apt-get command is with the following packages:
    - sudo apt-get install libgl1-mesa-dev libfontconfig1-dev libfreetype6-dev libx11-dev libxext-dev libxfixes-dev libxi-dev libxrender-dev libxcb1-dev libxcb1 libx11-xcb-dev libx11-xcb1 libxcb-glx0-dev libxcb-keysyms1-dev libxcb-image0-dev libxcb-shm0-dev libxcb-icccm4-dev libxcb-sync-dev libxcb-xfixes0-dev libxcb-shape0-dev libxcb-randr0-dev libxcb-render-util0-dev libjasper-dev libmng-dev libjpeg-dev libpng12-dev libtiff5-dev libgif-dev

3. Download Qt 5.5.1 and do a full install from this link. (NOTE: If you use a newer version of Qt, the android or iOS packages are not necessary).

4. Change directory where Qt source files are installed (In this case, Qt was installed at "/opt/qt/5.71")

    cd /opt/qt/5.71/Src

5. The "Makefile" has to be created using the "configure" application. The parameters passed to the "configure" application must be the name of the Qt libraries our application will need, accept the LGPL license, and the platform (compiler) to be used and the path where to install the new compiled version of Qt (in our case the path *"/usr/local/Qt-5.5.1*" will be used). The more libraries used, the more time it will take to compile. For gLAB GUI, with these parameters is sufficient (this will take a few minutes):

    sudo ./configure -static -opensource -release -confirm-license -no-compile-examples -nomake tests -qt-zlib -qt-libpng -qt-libjpeg -qt-freetype -qt-pcre -qt-harfbuzz -largefile -gtkstyle -qt-xcb -qt-xkbcommon -prefix "/usr/local/Qt-5.5.1"

    NOTES:
    - If we want to be able to have a debug version of the static build, we need to add the parameter "-debug".

- Optionally, you can add these parameters “-no-pulseaudio -no-alsa -no-qml-debug -skip qtwebkit -skip qtwebkit-examples” to skip the compilation of pulseaudio, alsa, QML and Webkit modules. In some Qt versions are automatically unselected, in others they are selected.

6. Compile Qt. With parameter “-j N” you can set the number of cores to used (N is the number of cores). Depending on the processing power and the number of cores, this process may take several hours to complete.

   sudo make -j 4

   NOTES:
   - No errors should occur, but sometimes some an error may occur on libraries you are not using (for example, sometimes an error with QML libraries occurred, which are not used in gLAB GUI). In this case, you can add the parameter *“--ignore-errors*” to ignore these errors when compiling. The list of Qt libraries used is available in the source file “glab_gui.h” (the “#include” directives with libraries that start with “<Q”).
   - If an error occurs compiling “webp” library, with the error “webp/encode.h” or “webp/decode.h” file not found, then you will need to download and copy the “webp” library to the specific Qt library by following this path:
     - Download “webp” library for linux from this link (in our case we are downloading file “libwebp-0.4.1-linux-x86-32.tar.gz”.
     - Uncompress the file with command “*tar –xvzf libwebp-0.4.1-linux-x86-32.tar.gz*
     - Go to the “include” directory: *“cd libwebp-0.4.1-linux-x86-64/include/”*
     - Copy the “webp” directory to the Qt source library: *“cp -r webp /home/gage/Qt5.5.1/5.5/Src/qtimageformats/src/plugins/imageformats/webp/”*

7. Install the new version of Qt in the folder specified in the “-prefix” parameter in step 5. As in step 6, this process may take several hours to complete.

   sudo make install -j 4

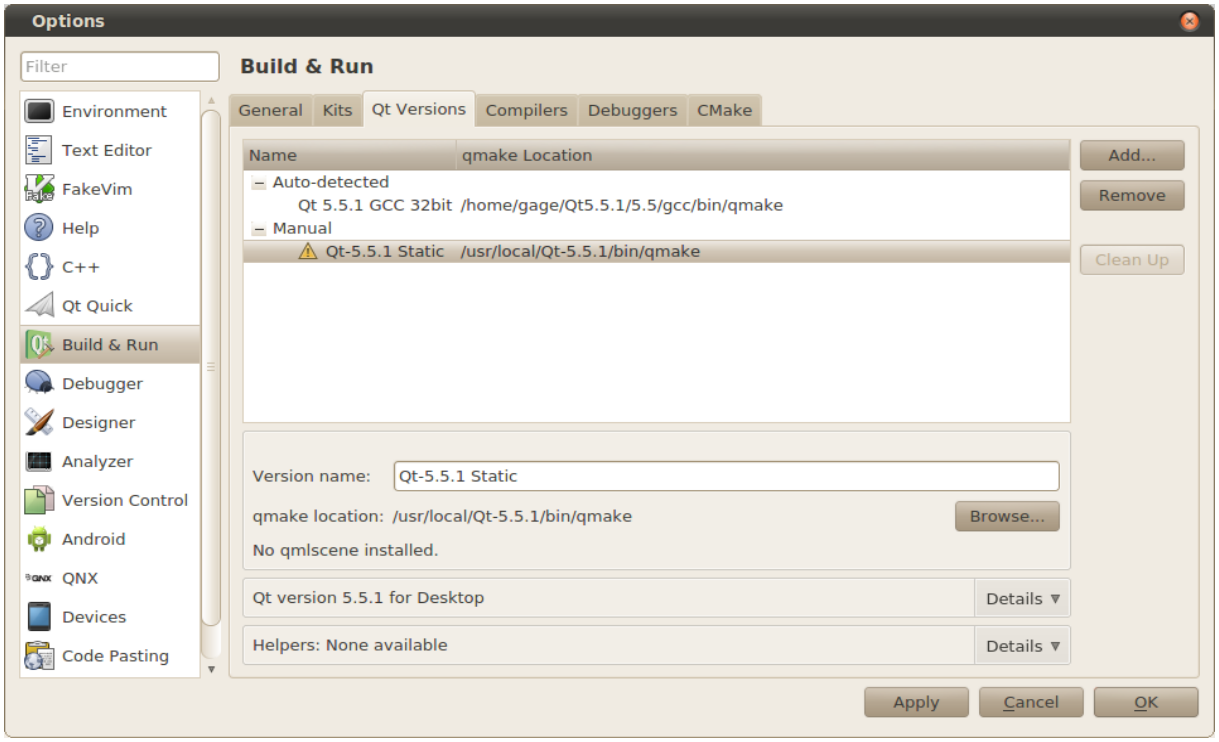
   NOTE: As in step 9, if errors occur in unnecessary libraries, add parameter *“--ignore-errors*”.

8. Clean the intermediate files created during compilation (this is just for saving disk space). **(If you want to make a new Qt build, this step is mandatory)**

   sudo make clean -j 4
   sudo make distclean -j 4

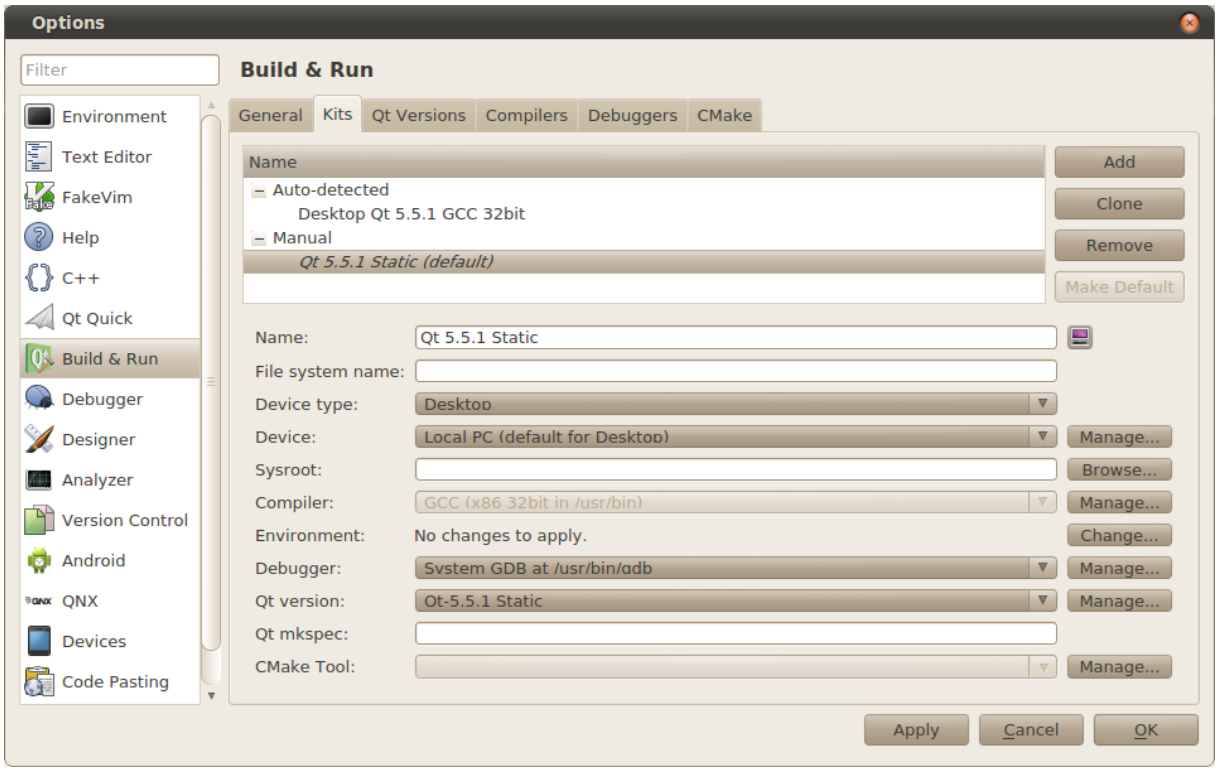
9. Open Qt creator, and open your project.

10. In Qt creator, the new Qt Static version has to be added:
    - Go to Tools→Options.
    - In the left pane, select “Build & Run”.
    - Select the “Qt Versions” tab.
    - Click in the “Add” button, and search for the “qmake” file, which will be in the “bin” folder where the static Qt was installed. In this case, it should be in ““/usr/local/Qt-5.5.1/bin" folder.
    - Set a name for this version (any name is valid).
    - Click in “Apply” in the bottom of the window.
    - The result should look like this:



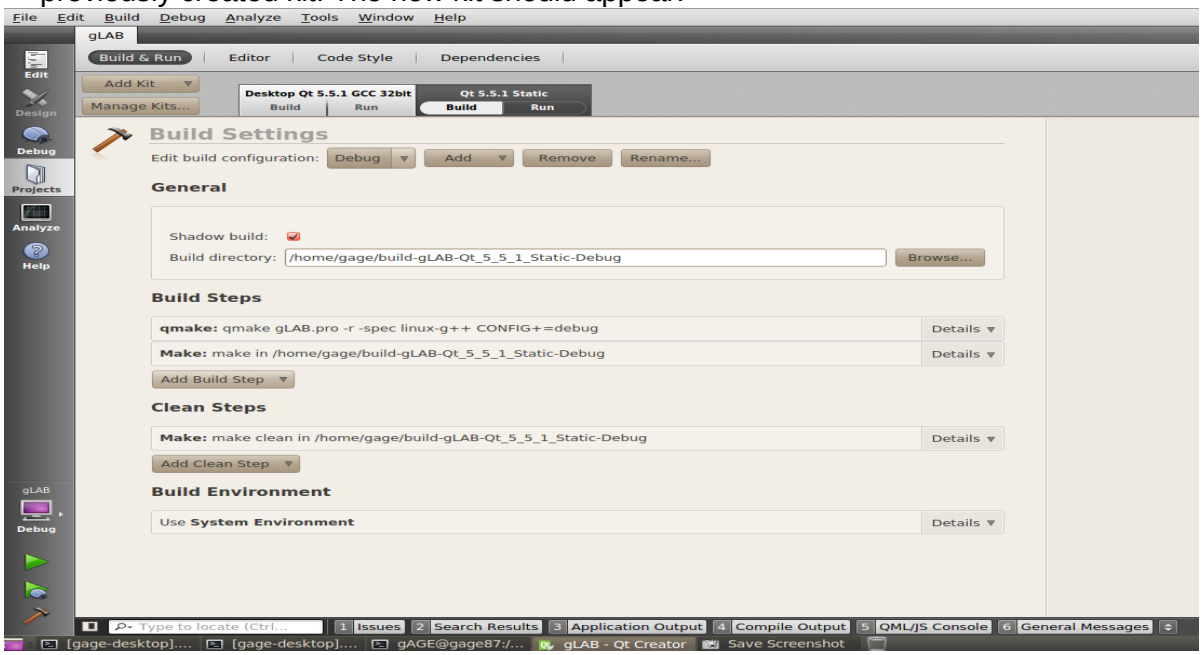
Ignore the warning, as it is due to “the compiler may not produce valid code”, which is false, because we are using the same compiler as the Qt company used for building this Qt version.

11. Add a new build kit with the static Qt version:
    - Click in the “Kits” tab.
    - Click in the “Add” button
    - Set a name for the kit (any name)
    - In the compiler section, set GCC for C and C++
    - In the Qt version, select the one created in step 10.
    - The result should look like this:

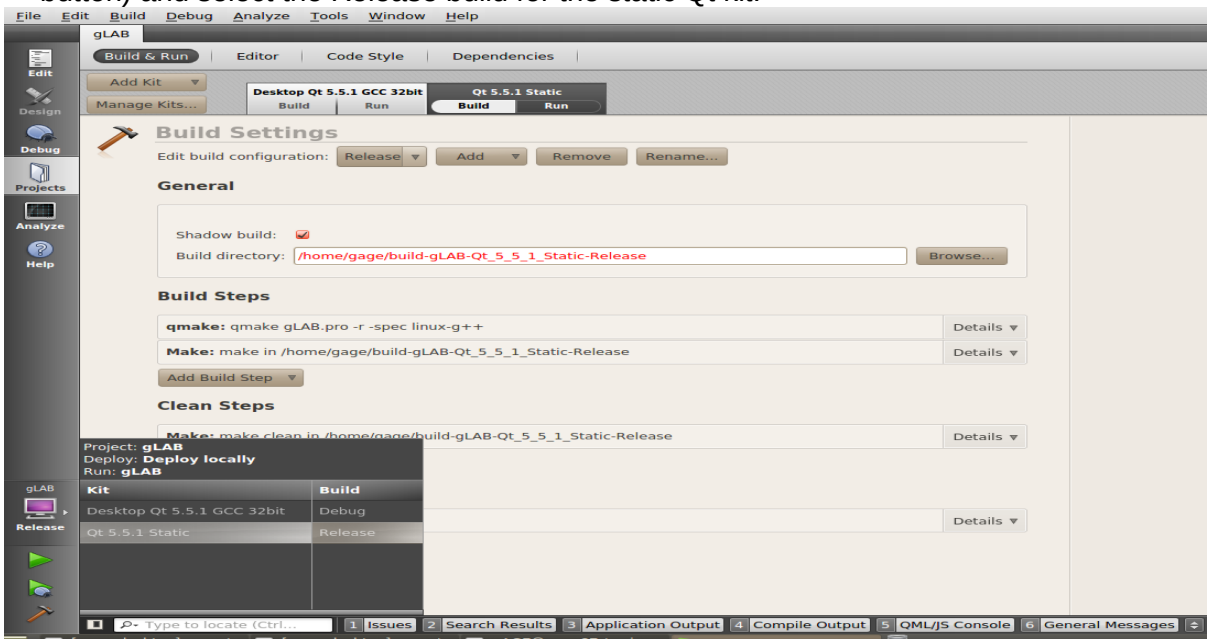


12. Click in “OK” in the bottom part of the window.

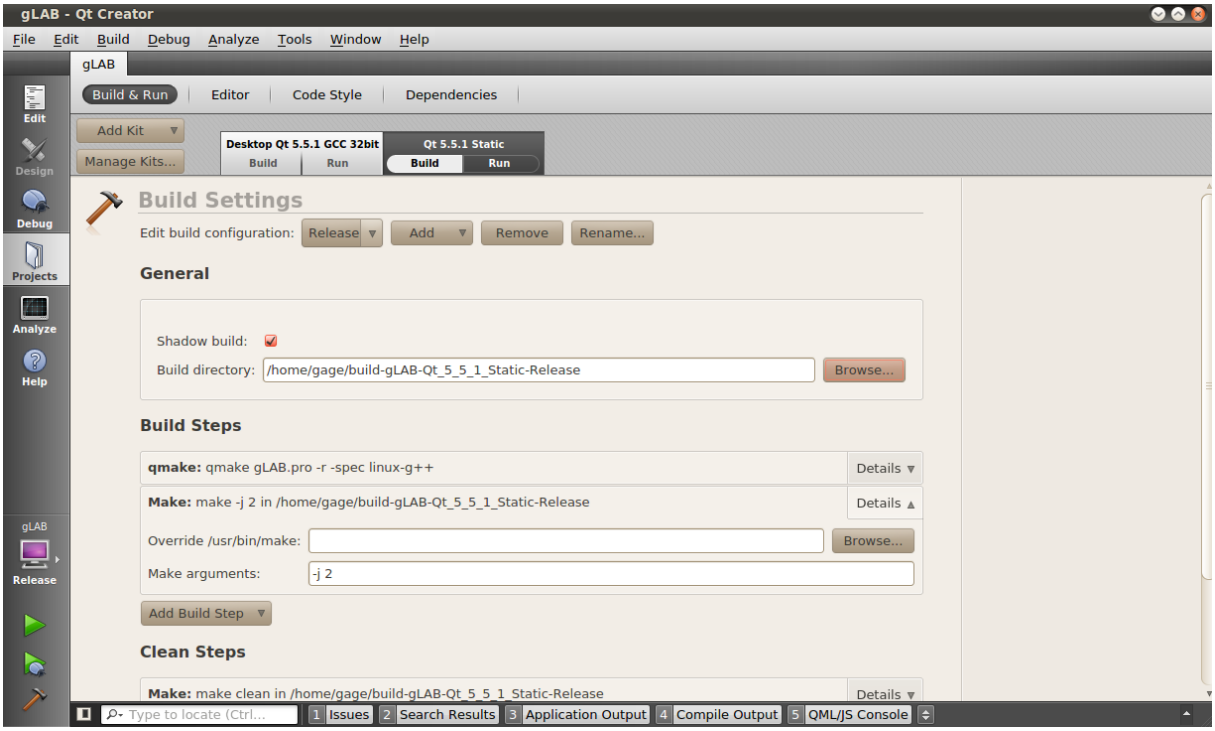
13. In the left pane, click in the “Projects” button. Click in the “Add kit” button and select the previously created kit. The new kit should appear:



14. The new kit is ready to build, but we need to make sure we select the Release build, as the Debug build will not be available (unless the “-debug” parameter was provided in step 5). To select the Release build, click in the button with the monitor icon (over the play button) and select the Release build for the static Qt kit:



15. Optional: To compile with more than one CPU, click in the “Projects” button, select the “Build” line of Static ARM Qt Kit, then in the right side, below the “Build Steps” section, click in the “Details” button of the “Make” line. The line with “Make arguments” will appear. In this line, write “-j 2” (or substitute 2 by the number of CPUs used).



16. Optional: When compiling with a target machine of Ubuntu 10 to 16, the system library “libjasper” is linked, but it is not available any more in Ubuntu 18. To force this library to be statically linked, the Makefile created by Qt Creator has to be modified. A new Makefile will be created to avoid Qt Creator overwriting the modified file.

    - Open a terminal (with the default user)

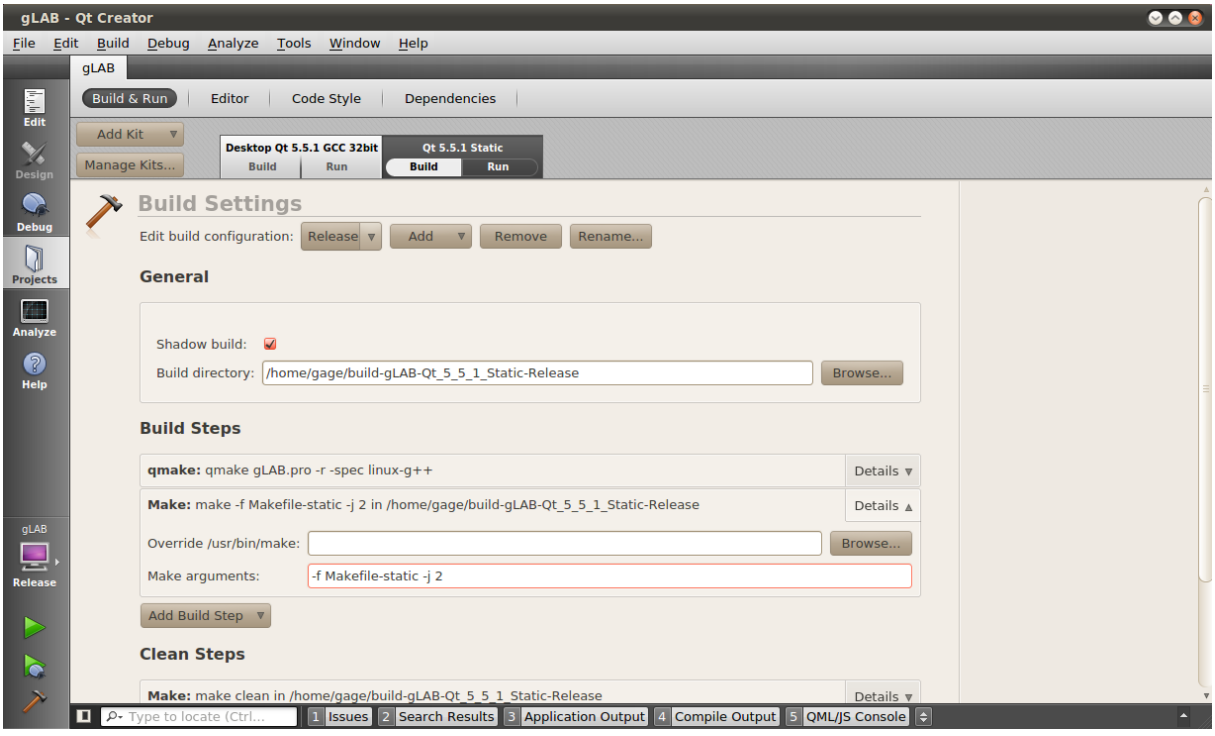
        - Change to the build directory created by Qt Creator (it is shown in the “Build directory” in the “Build Settings”, as in images on steps 13, 14 and 15). In this example it is “/home/gage/build-gLAB-Qt_5_5_1_Static-Release”

      cd /home/gage/build-gLAB-Qt_5_5_1_Static-Release

    - Add the flag for libjasper for forcing to be statically built. To do this, the flag “-ljasper” has to be replaced for “-Wl,-Bstatic -ljasper -Wl,-Bdynamic” with the following command:
      sed -e 's/-ljasper/-Wl,-Bstatic -ljasper -Wl,-Bdynamic/' Makefile > Makefile-static

    - As in step 33, in the “Make arguments”, add at the beginning (prior to the “-j” parameter, if provided) the text “-f Makefile-static”

    - The result should look like this:



17. Compile the application with the “Play” button.

18. To test the that the executable does not have any Qt dependency, go to the folder where the binary is created (in our case, as we can see in the screenshot above, the binary will be in the path “/home/gage/build-gLAB-Qt_5_5_1_Static-Release”), execute the following command:

    ldd gLAB_GUI | grep –i qt

    No output should appear (that is, Qt library dependencies)

End of Document