

## NxNandManager : Set up and build project with Qt (GUI)


Intel 64 Processor - Windows - MinGW64

2022, January

### Pre-requisites :

Download Qt (open source) : <https://www.qt.io/download>

During the installation, make sure to install at least QT 5.13+ for MinGW

- 
- A screenshot of the Qt installer window showing the selection of components. The 'Qt 5.13.2' section is expanded, and the following components are listed: WebAssembly, MSVC 2015 64-bit, MSVC 2017 32-bit, MSVC 2017 64-bit, MinGW 7.3.0 32-bit (checked), MinGW 7.3.0 64-bit (checked), UWP ARMv7 (MSVC 2015), UWP x64 (MSVC 2015), UWP ARMv7 (MSVC 2017), UWP x64 (MSVC 2017), and UWP x86 (MSVC2017).
- ☐ Qt 5.14.2
  - ☒ Qt 5.13.2
    - ☐ WebAssembly
    - ☐ MSVC 2015 64-bit
    - ☐ MSVC 2017 32-bit
    - ☐ MSVC 2017 64-bit
    - ☒ MinGW 7.3.0 32-bit
    - ☒ MinGW 7.3.0 64-bit
    - ☐ UWP ARMv7 (MSVC 2015)
    - ☐ UWP x64 (MSVC 2015)
    - ☐ UWP ARMv7 (MSVC 2017)
    - ☐ UWP x64 (MSVC 2017)
    - ☐ UWP x86 (MSVC2017)

### Set up & build project :

My installation folder is S:/dev but you should choose your own.

First of all, clone NxNandManager's git repo inside installation folder :

```
elibo@DESKTOP-AM14A6I MINGW64 /s/dev
$ git clone https://github.com/elibo/NxNandManager
Cloning into 'NxNandManager'...
remote: Enumerating objects: 3089, done.
remote: Counting objects: 100% (894/894), done.
remote: Compressing objects: 100% (637/637), done.
remote: Total 3089 (delta 611), reused 517 (delta 253), pack-reused 2195
Receiving objects: 100% (3089/3089), 33.47 MiB | 19.52 MiB/s, done.
Resolving deltas: 100% (2081/2081), done.
```

Now, download OpenSSL's pre-compiled binaries for MinGW :

[https://www.elibo.com/OpenSSL\\_mingw\\_build.rar](https://www.elibo.com/OpenSSL_mingw_build.rar)

Extract the content of the archive in the installation folder.

This should be the content of your installation folder :

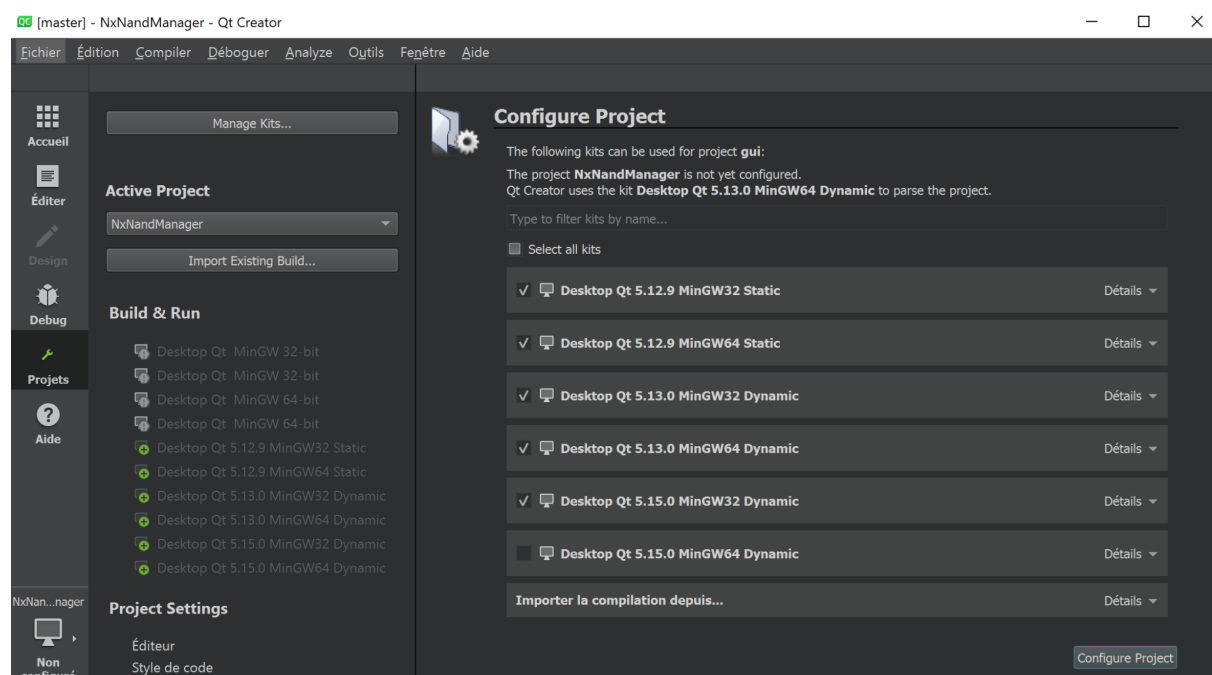
```

elibo@DESKTOP-AM14A6I MINGW64 /s/dev
$ ll
total 36636
drwxr-xr-x 1 elibo 197611      0 janv. 25 10:47 NxNandManager/
-rw-r--r-- 1 elibo 197611 37509969 janv. 25 11:23 openssl_mingw_build.rar
drwxr-xr-x 1 elibo 197611      0 août 10 2019 openssl_mingw32/
drwxr-xr-x 1 elibo 197611      0 août 10 2019 openssl_mingw64/

```

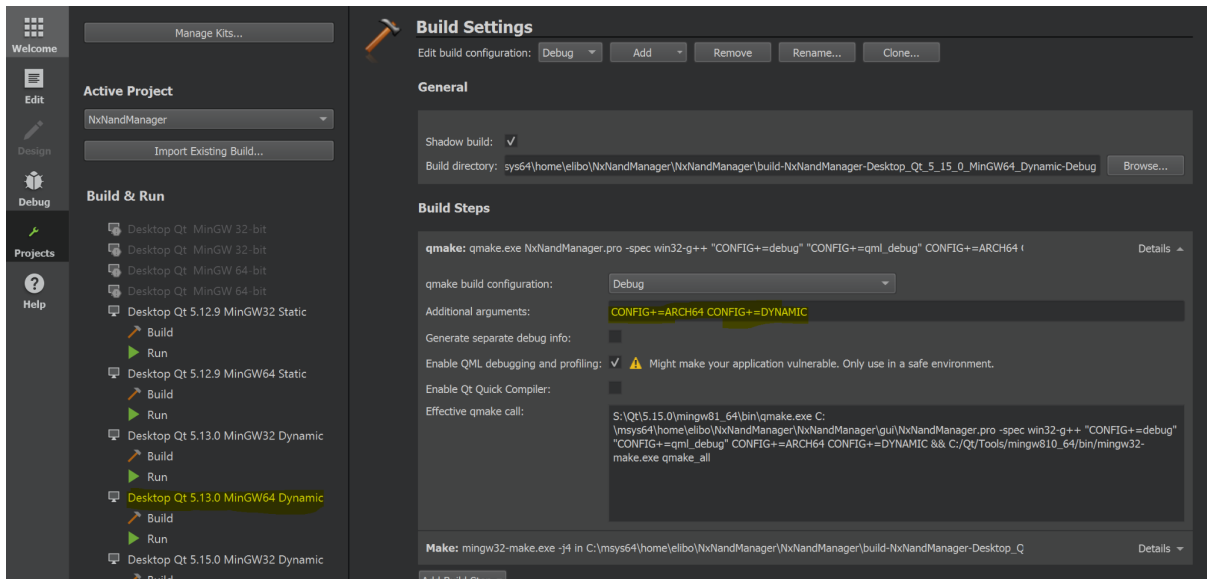
Open “QT Creator” **as an administrator (important)**, then open project’s file :  
 S:\dev\NxNandManager\NxNandManager\gui\NxNandManager.pro

The first time the project is opened, you’ll have some config to do. You should select every Qt kit you want to build the project with. For the purpose of this tutorial, you should at least select MinGW64 5.13+

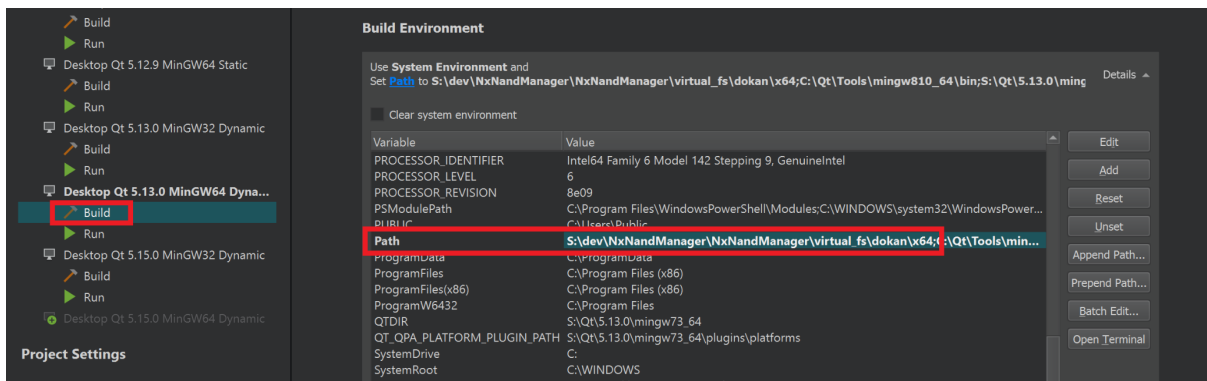


**NB :** I have custom kits to build statically but in your case you shouldn’t see any mention of “Static” or “Dynamic” in your kit list.

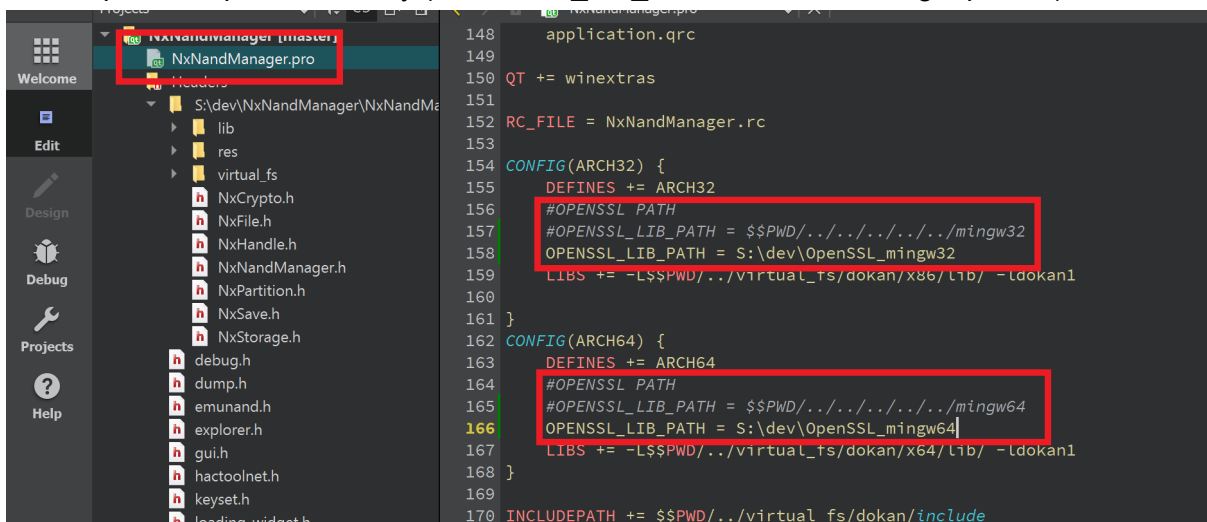
From “Projects” tab, select “Build” from your desired kit.  
 Add extra arguments in build settings Settings : CONFIG+=ARCH64 CONFIG+=DYNAMIC



Add the path to dokan library to environment variable PATH (in my case S:\dev\NxNandManager\NxNandManager\virtual\_fs\dokan\x64) :

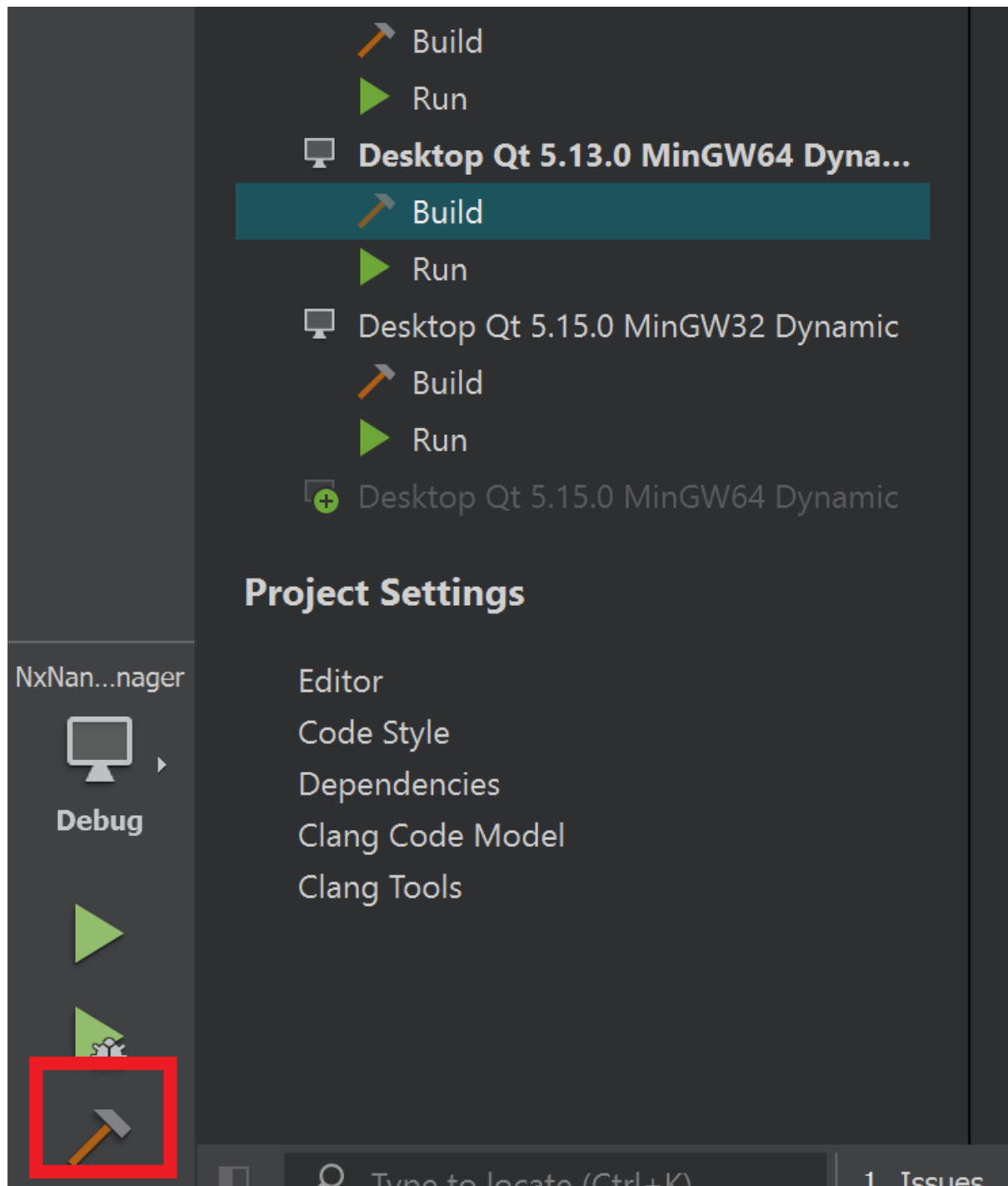


Set the path to OpenSSL library (OPENSSL\_LIB\_PATH, NxNandManager.pro file)



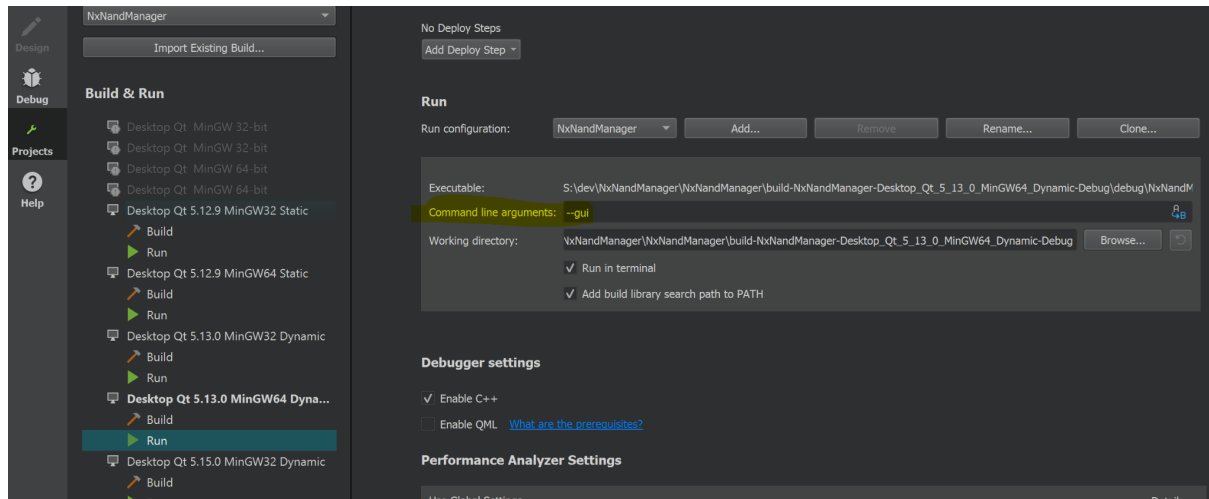
Configuration is done. You won't have to do it next time.

Now build the project :



**Run :**

Go to "Projects" then select MinGW64 Kit "Run" and add `-gui` argument :



You can now run the program :)

## **Deploy :**

Default Qt framework is build dynamically so you'll need to deploy the project if you want to run the program outside Qt's environment (Qt Creator), a.k.a put all the required DLL's in the same folder as your executable :

- Either use Qt "Windows Deployment Tool" : <https://doc.qt.io/qt-5/windows-deployment.html>
- Or grab all the \*.dll files inside this [previous build of NxNandManager](#) (and put them in the same folder as NxNandManager.exe.

Don't forget dokan1.dll (not in the archive, you can grab a copy [here](#))

## **Troubleshooting :**

Worth remembering :

- Qt creator, if not run as administrator, will refuse to run your build!
- If GUI is not launching but CLI is, check the "ENABLE\_GUI" variable in gui.h file.
- If compilation ends up with no errors (warnings are fine) but you can't run the executable inside Qt Creator it's probably a linking issue. Check the required DLLs or your PATH environment variable. You'll need DLLs for Qt, OpenSSL & Dokan.

## **Static build :**

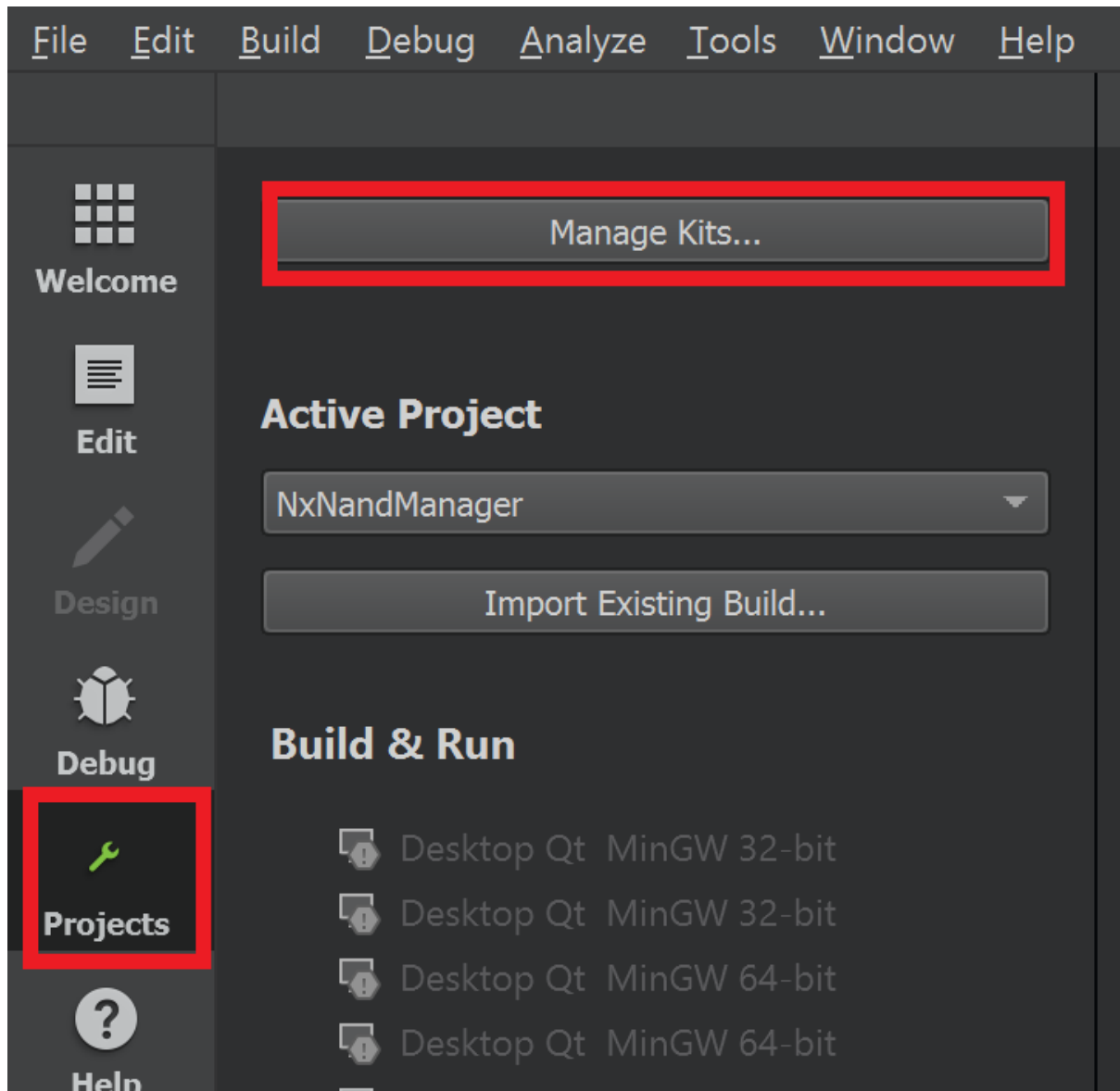
If you want to get rid of all the DLLs (except dokan2.dll) you'll need a static build of Qt. An official Qt kit is a collection of pre-built libraries to be linked dynamically (DLLs). So you would have to re-compile all Qt source code to build a static version of every lib.

As this process can be very long, the easiest way to go is to download my own pre-built version of Qt 5.12.9 : <https://eliboa.com/QtStatic 5.12.9 for MinGW.rar>

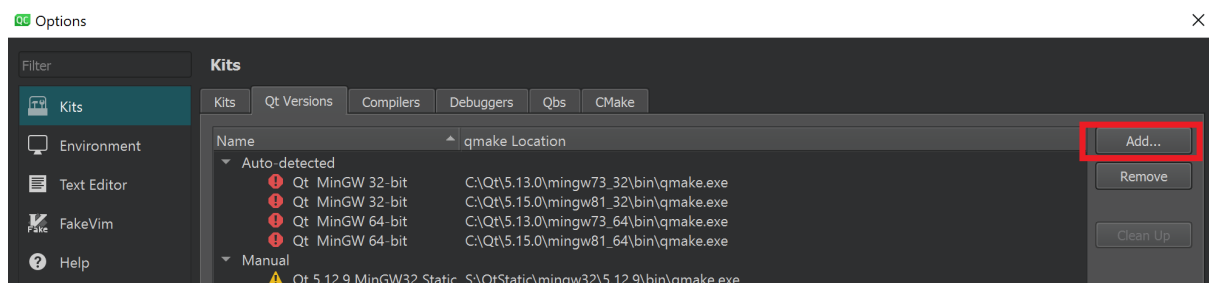
Extract the archive to "C:/" (S:/ in my case).

You should see a folder named C:/QtStatic

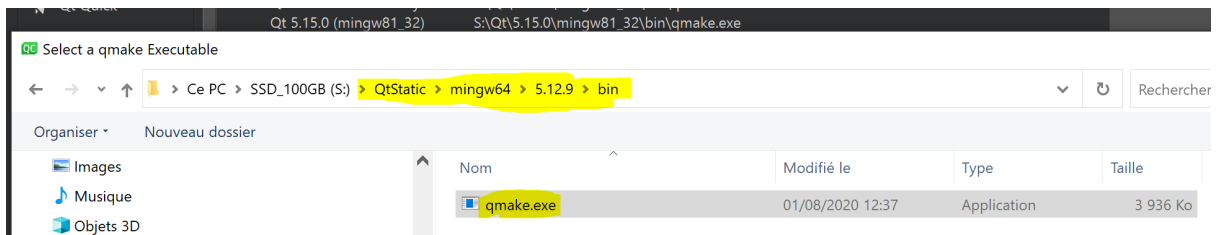
In Qt Creator, from the “Projects” tab select Manage kits...



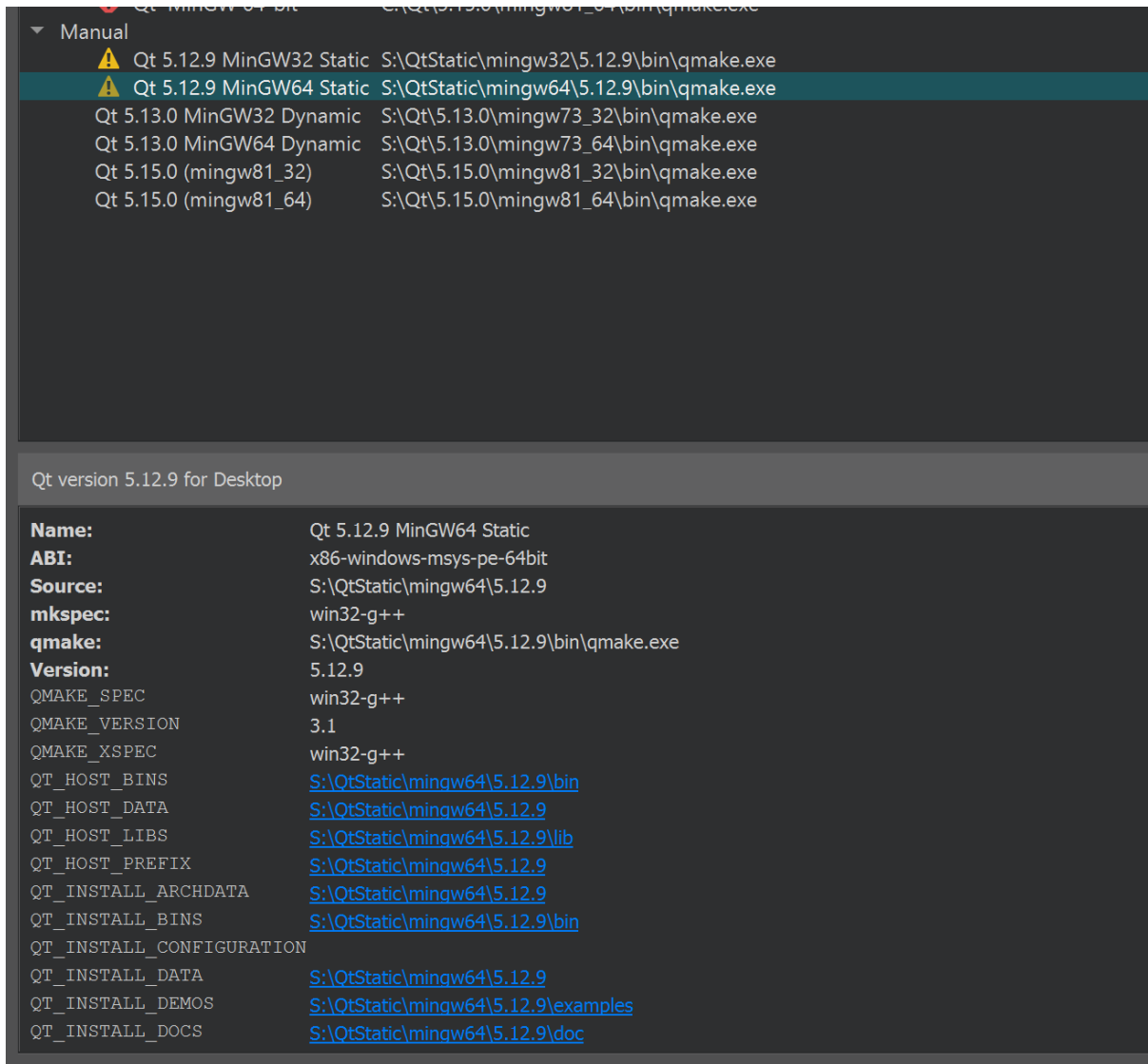
Navigate to “Qt Versions” tab. Add a new Qt (static) version for mingw64 :



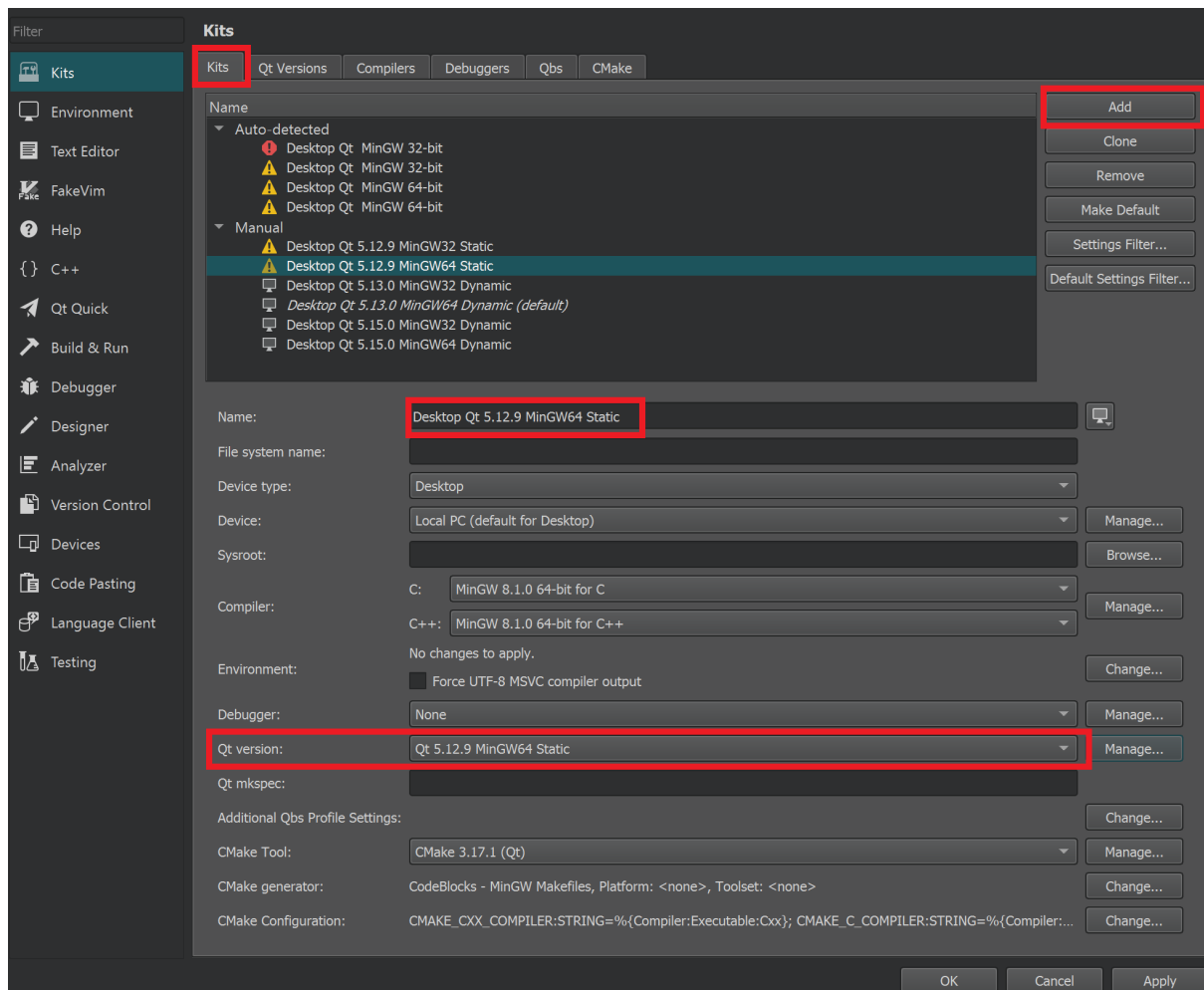
select “qmake.exe” file from C:/QtStatic/mingw64/5.12...



You should see something like this once added :

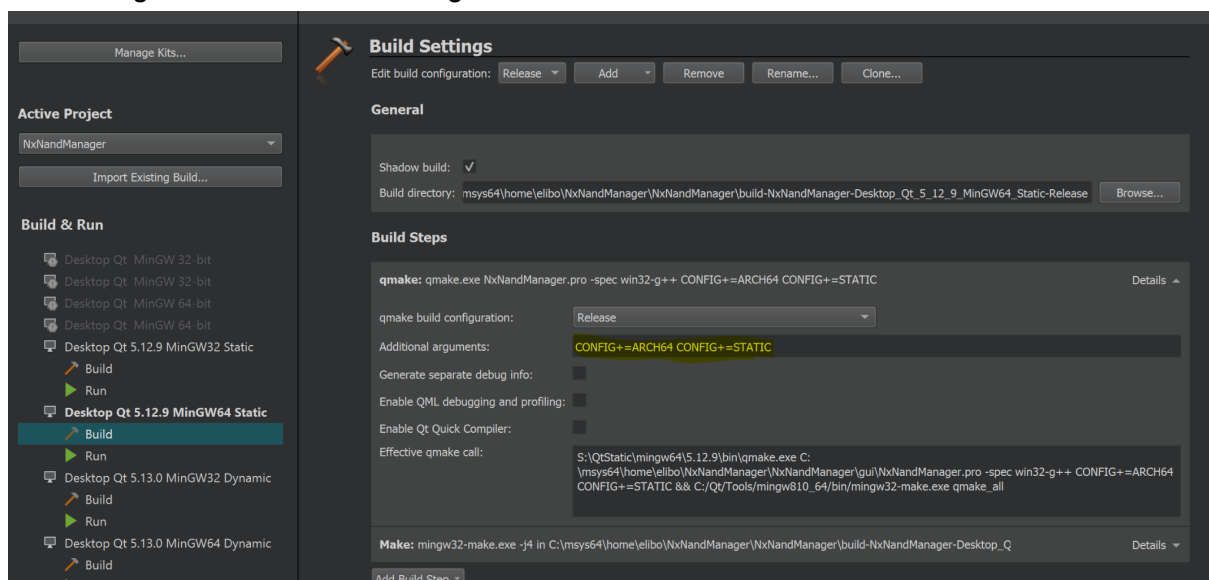


From the “Kits” tab, add a new kit for your Qt static version :



Apply & exit Options.

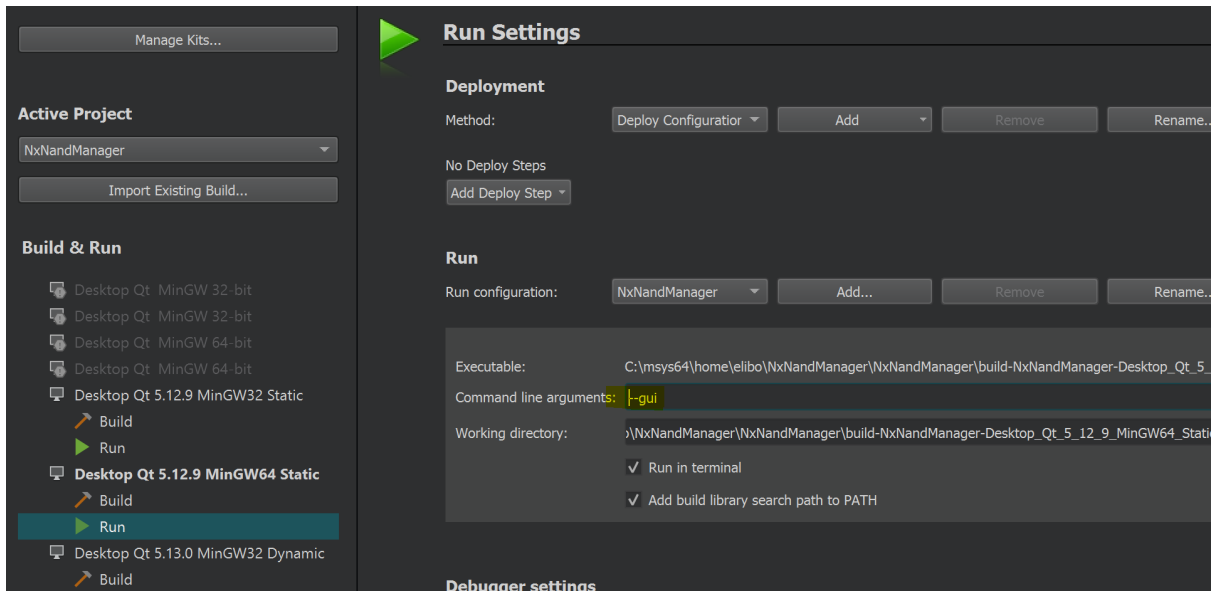
Now configure build and run settings :



**NB :** Don't forget to add path to Dokan dll's folder in PATH env. variable (same as dynamic config)

Run settings :





You should now be able to build.