SAS Scatter2

Docker usage documentation

Michael Wagener, JCNS-1

August 4, 2025

Contents

Table 1: Document revision history

Date	**Short description**
04. Aug 2021	Start.

Snapshots on https://github.com/neutron-simlab/CrystalScatter with global access

1 Preface

During the development of the Chatbot with a special command line interface to the sas_scatter2Cons program, this program must be available from everywhere. So the best way is to put it into a docker image.

In this short documentation all steps are described to generate this docker image.

2 Base image

The sas_scatter2Cons program is written in C++ with the Qt library. So, the first base docker image must be one with Qt 6 support. In the docker hub I found the images *g76r/qt6-builder* and *g76r/qt6-runner* with the version Qt 6.8.3 and a debug option.

The builder image contains everything to compile the source and generate the executable needed. The runner image contains only the Qt runtime libraries.

3 Do the compilation

For the compilation we start a docker container based on the builder image and do the compilation. Inside this docker container we generate a virtual directory pointing to a real directory on the host system. This directory contains the complete source code. All intermediate files and the executable are generated also here. This docker container will not be saved because it is not needed later.

```
docker run --rm -v "%cd%:/home/user/project" g76r/qt6-builder:qt-6.8.3-debug \
       sh -c "cd /home/user/project; rm -rf build; mkdir build; cd build; \
       qmake ../sas_scatter2Cons.pro; make"
```

The parameters for the *docker run* command are:

- --rm
  This container will be deleted after exit
- -v "%cd%:/home/user/project"
  Is the virtual directory: %cd% means the current directoy (Windows notation) and can be changed to the absolute source path, the part after the colon is the path inside the docker container
- g76r/qt6-builder:qt-6.8.3-debug
  Is the image name to run
- sh -c
  Opens a shell and perform the following commands:
  * cd /home/user/project - *go to the source directory on the host*
  * rm -rf build - *delete the old build directory*
  * mkdir build - *make the build directory*
  * cd build - *go to this build directory*
  * qmake ../sas_scatter2Cons.pro - *generate the makefile*
  * make - *compile the source and generate the executable*

  Some warnings from the compilation can be ignored. These parts are under development and they have no impact to the execution.

4 Generate the runtime image

The generation of the runtime image consists of multiple steps.

First run a temporary container, install two libraries and copy the executable from above into the containers filesystem:

```
docker run -t --name scattercons-tmp  -v "%cd%:/home/user/project" \
       g76r/qt6-runner:qt-6.8.3-debug sh -c "apt install -y libxkbcommon-dev \
       libgl-dev; cp /home/user/project/build/sas_scatter2Cons /bin"
```

The parameters for the *docker run* command are:

- -t
  Generates a tty terminal connection, so you can see progress bars (can be omitted)
- --name scattercons-tmp
  This container must be named to be referenced later
- -v "%cd%:/home/user/project"
  Is the virtual directory and must be the same as above for the compilation
- g76r/qt6-runner:qt-6.8.3-debug
  Is the image name to run
- sh -c
  Opens a shell and perform the following commands:
  * apt install -y libxkbcommon-dev libgl-dev - *install two libraries*
  * cp /home/user/project/build/sas_scatter2Cons /bin - *copy the executable into the container*

Second remove the old destination image and generate it again with the same name:

```
docker image rm crystalscattercons-run
docker container commit -m "ConsExec" scattercons-tmp crystalscattercons-run
```

The parameter -m with the text is the commit message for documentation inside the image layers and can be changed. If you want, you can add the author with -a "...".

Last, remove the temorary container:

```
docker container rm scattercons-tmp
```

The resultant image (here named *scattercons-run*) can be published to docker hub.

5 Use the runtime image

The generated image can be used to calculate some CrystalScatter images. In the following example the host path is specified absolutly to my development file structure and some Chatbot parameters are used:

```
docker run --rm -v "C:\SimLab\CrystalScatter\Mcp:/home/user/project" \
       crystalscattercons-run sh -c "cd /home/user/project; sas_scatter2Cons \
       --mcpinp testmcppar.txt --mcplog testdocker.log \
       --mcpimg testdockerout.png"
```

The parameters for this *docker run* command are:

- --rm
  This container will be deleted after exit
- -v "C:\SimLab\CrystalScatter\Mcp**:**/home/user/project"
  Is the virtual directory (Windows notation in this example)
- crystalscattercons-run
  Is the image name to run
- sh -c
  Opens a shell and perform the following commands:
  * cd /home/user/project - *go to the directory on the host for the data*
  * sas_scatter2Cons - *start the executable*
  * --mcpinp testmcppar.txt - *input parameter file in Chatbot notation*
  * --mcplog testdocker.log - *Logfile, can be omitted*
  * --mcpimg testdockerout.png - *output image filename*

If you want to see all possible parameters fo the sas_scatter2Cons program, use:

```
docker run --rm crystalscattercons-run sh -c "sas_scatter2Cons --help"
```