Running SU2 in a Docker container

Tommaso Bellosta

September 26, 2023

Introduction

This short note will guide you through the installation of a docker container to run SU2 and gmsh on a Windows system. This is just one of the many ways that SU2 can be run on a PC. The preferred way, in terms of performance, is to compile SU2 directly on your PC. Linux is the preferred environment (i.e. most tested/supported) but a short guide is also available to compile SU2 also on macOS and Windows. Having a Linux environment on you PC does not require the user to delete its current OS, but can be installed alongside yor preferred OS on a separate partition of your disk. A quick search on google will provide all the resources to perform such a task. On a Windows system you can also use WSL which will give you access to a Linux shell directly on you PC (from wich you can compile SU2 following the build instruction for Linux). **We will not provide any support for problems you might have during partitioning/installing linux and compilation on macOS/Windows. So, do so if you are already accustomed to compiling code on your OS/installing linux, or you are confident enough in your IT skills.** We will however provide limited support if you chose to run the code via a virtual machine as shown during the lecture and explained in the pdf guide avalable on webeep. Remember to install `ubuntu20.04` if you want to be able to follow the guide step by step without problems. In later versions of ubuntu, the openMPI package available with `apt-get` is not compatible with SU2, and you would need to manually build openMPI before installing SU2 (again, no support provided by us). A second, partially supported way, is to run SU2 via a docker container (i.e. a slimmed down virtual machine) on Windows. Using this approach, no compilation is needed. The only sytem requirement is to install Docker Desktop for Windows available at this link. You will get a fully functioning Linux shell from which you can directly run SU2 and gmsh, but no dektop environment.

Installing Docker Desktop

Docker on Windows will leverage the existing WSL or Hyper-V backends to run containers. This guide was tested using the WSL backend, so this is the prefered choice. First, check if your system meets the requirements listed here, otherwise update your Windows version. Then, if not done before, install the WSL on you pc. Run PowerShell as an administrator and execute the following command:

```
$ wsl --install
```

Follow the instructions that will be displayed (if any) and complete the installation of WSL. Reboot your system if asked to. Then download and run the Docker Desktop installer available at this link. Follow the instructions, and when prompted select the WLS2 backend (it should already be the default option if WSL is installed on your system).

Create and run the SU2 docker container

Launch the Docker Desktop executable. You should be prompted to allow the Docker Engine process to make changes to your system. You must accept the prompt otherwise Docker will not be able to run your container. Once initialized, Docker Desktop will show you the available images (e.g. iso filed of ubuntu) containers (e.g. your virtual machines) and volumes

(used to save container data to make it avalable also is you decide to delete your container, we will not use this). Everything should be empty, we still haven't created anything. Before creating the actual virtual machine (container), create a folder on your user directory and call it `dockerData`. It's path should be something like `C:\Users\yourUsername\dockerData`. Copy it, as it will soon be needed. If your user name contains spaces, please put it in between apices (i.e. `C:\Users\'User name'\dockerData`). This folder will be shared between the host (your Windows system) and the container (the SU2 virtual machine). Data written in this folder is visible from both host and container and it will be used to share data.

Once docker Desktop is running, open a new window of PowerShell and execute the following command:

```
$ docker run -it -v C:\Users\yourUsername\dockerData:/home/cfd/DATA --name su2-docker tbellosta/su2:v8_ubuntu23.04
```

If the command returns an error comlaining about missing permissions to pull the image, create a free account on docker-hub and login with your credentials on the Docker Desktop app. Alternatively, you can also login on the PowerShell with the command:

```
$ docker login
```

and enter your newly created docker-hub credentials. After a while (is needs to download a modified ubuntu image opf less than 2GB), the container will be created and the power shell window should display your session on your new container. You will see something like: `cfd@f234hj23blabla:~$`. You are now logged into your container as user cfd and can play around in your new virtualized linux system. In the home directory you should see two folders `Code` and `DATA`. `Code` contains the compiled SU2 executables, together with a quick start testcase, and a folder containing the configuration files of all SU2 test cases, plus other software that is needed by SU2 (i.e. openMPI). To start running SU2, go to the QuickStart directory and run the test case:

```
$ cd Code/SU2/QuickStart
$ SU2_CFD inv_NACA0012.cfg
```

You can also run SU2 in parallel with the command:

```
$ mpirun -np 2 SU2_CFD inv_NACA0012.cfg
```

where 2 is the number of cores that will run the computation. The resources available to your container (e.g. number of cores available, amount of RAM) are handled by your WSL configuration. Please, check on google on how to modify these WSL settings if needed. Folder `DATA` is the shared folder with your host system. It is just the mount location of the `C:\Users\yourUsername\dockerData` folder you have on Windows. Everything you create in the container outside of this folder remains available untill you delete the container but is only accessible by the container and not your Windows system (You can copy files and folder between host and container with the command `docker cp` to be run from the host). Files in folder DATA are instead available also on your host in `C:\Users\yourUsername\dockerData`. You can use this folder to share data between host and container. For example, you can run your simulation in any folder in your container (e.g. in `/home/cfd/NACAeuler`), copy the results in `/home/cfd/DATA` and visulize them from paraview in Windows.

To stop the container from running, execute:

```
$ exit
```

Next time you want to start your container, simply execute the following command from PowerShell:

```
$ docker start -i su2-docker
```

As always, you need first to launch the Docker Desktop executable and if asked allow the Docker Engine to run. Remember to always close the container after you are done with the `exit` command. If you forget to do so, the container will continue running and consume the host's resources. to chek if this is the case, open PowerShell and run:

```
$ docker ps -a
```

If the status of the container is running, you can stop it with:

```
$ docker stop su2-docker
```

As stated in the introduction, partial support is provided if you decide to follow this guide. In case of problems, plase always try to solve them independently with the help of the online documentation (and the power of google, stack overflow, chatGPT). If that's not enough you can write on the weeBeep forum, so that others in need may benefit from the support we will provide (or at least try to do so).