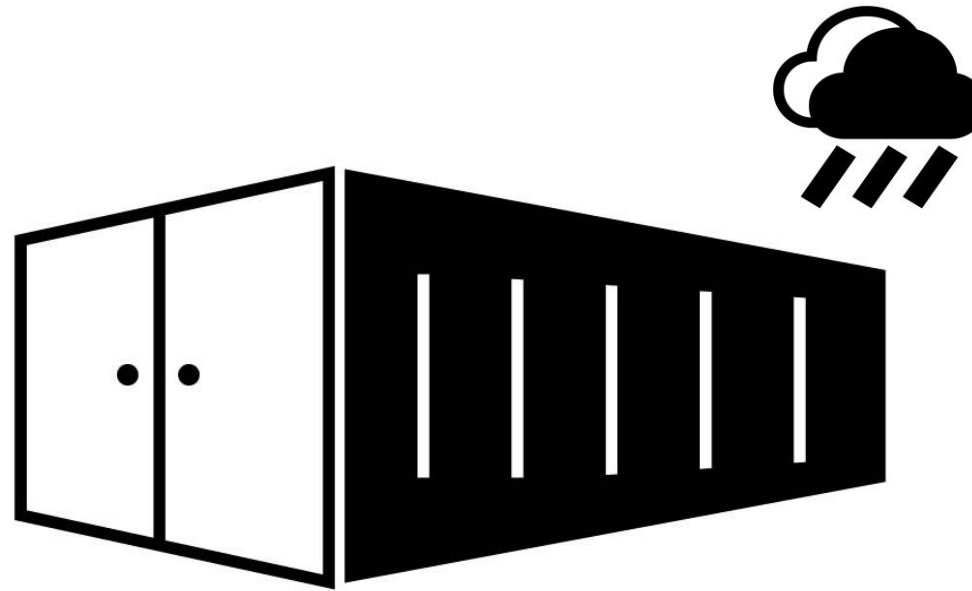




Tech-Talk: Containerizing atmospheric models

A simple guideline why to use containers



Agenda

- Initial Situation
- Containers/Singularity
- Example 1: ICONIC – ICON in the cloud
- Example 2: GFDL SHiELD v2021b in a container
- Hand's On: Build our own container
- Outlook: Container for ICON-ESM
- References

Initial Situation: Motivation

- Often, porting (climate) models can be quite complex. To compile such models on HPC's the system has to provide both a compiler and the necessary modules or libraries (`netcdf`, `hdf5`, `zlib`, ...). Moreover we use kind of complicated software stacks that are very host specific.
- Setting up the computing environment required by different models can present a challenge when running the model on a new hardware infrastructure or operating system. For example the model does not compile on the operating system due to incompatible libraries or issues with the native compiler.

Initial Situation: Our Problems

1. Compiling MPI-ESM using „buggy“ version of `intel-mpi`:

→ Running of model becomes really slow.

2. Using ICON-ESM on multiple nodes on VSC-4:

→ In some setups, the model hangs in different timesteps.

3. OS-Update on VSC-4 (software stack installation):

→ Essential modules for the use of ICON are not available at the moment.

These are only a few examples which are showing that there will be an urgent need to containerize the entire workflow.

Containers: Historic View

Before containerization

- Goods must be loaded and unloaded **individually**, many times by hand
- **Inefficient** – more time spend loading and unloading goods than transporting them
- **Insecure** – goods must be stored and handled by intermediaries during transport; potential loss and theft of goods

After containerization

- **Standardized** – containers of known dimensions and permissible weight tolerance
- **Efficient** – portable containers allow fast loading and unloading from multiple modes of transportation
- **Secure** – goods remained stored within the same container

Containers: Simple Explanation

- Containerization refers to packaging one or more applications into a container in a portable manner. It packages not only applications but all their dependencies, such as code, runtime environment and libraries, so the applications can run directly from one computing system to another.
- Unlike a virtual machine that emulates a whole computing system for use at the hardware layer, a container uses the kernel of the host machine and packages only the necessary components required to run applications. As a result, a container is lightweight and fast.

Singularity

- Singularity is a container platform, which allows you to create and run containers that package up pieces of software. One can build a container using Singularity on a laptop or a virtual machine and then run it on many of the largest HPC clusters in the world, in the cloud or on a simple workstation.
- The container itself is a single file, with the advantage that you don't have to worry about how to install all the software you need on each different (operating) system.
- Nowadays many container platforms are available (e.g. Docker), but Singularity focusses on: Verifiable reproducibility, integration over isolation, mobility of compute as well as a simple and effective security model.

Example 1: ICONIC – ICON in the Cloud

- The ICONIC project deals with the implementation of limited area version numerical weather prediction model (ICON-LAM) as container images and their execution as cloud software.
- **Containerization:** based on so-called images which are complete in itself and pre-configured ready to use.
- **Infrastructure as a Service (IaaS):** Cloud vendors (e.g. AWS, MS Azure, ...) support software containers and offer computing, storage and network resources.
- **ICONIC use case:** model area in Central Asia (countries with low capacities)
- **Cost estimates** (for two forecast runs each day): 100 USD/month for a 24/7 login node and less than 30.000 USD/year for e.g. 16 compute nodes

Example 1: ICONIC workflow

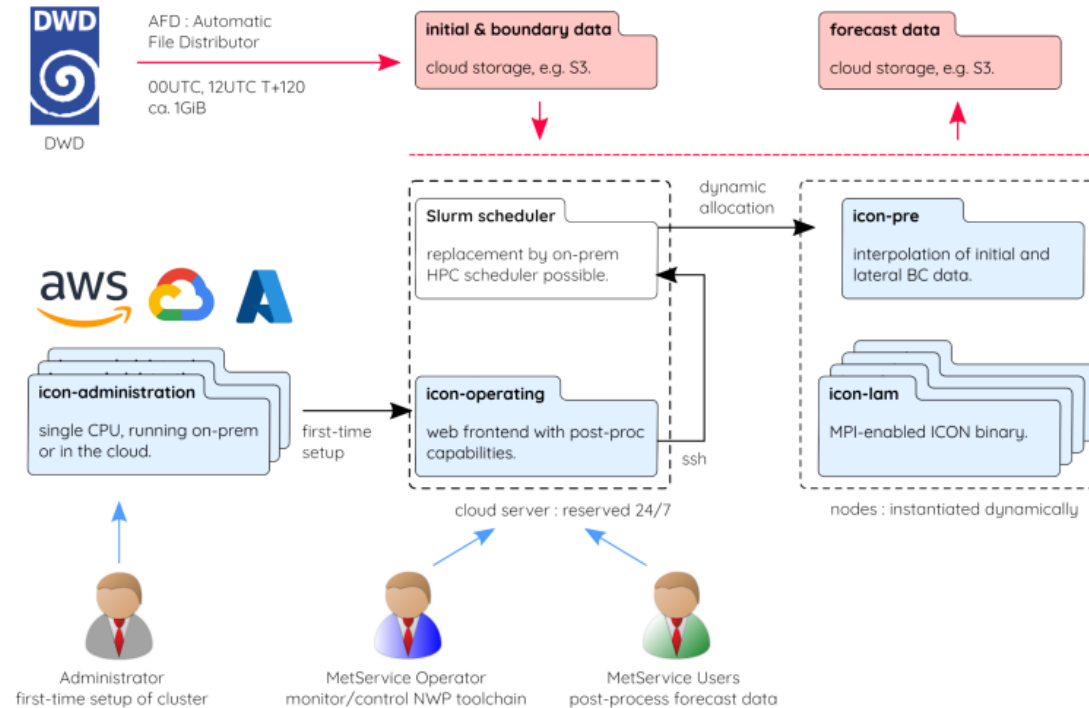


Figure 1: Schematic of the ICONIC workflow, containing the process participants, the services, and the data flow.

Example 2: GFDL: SHiELD v2021b in a container (1)

- SHiELD (System for High-resolution prediction on Earth-to Local Domains) is a unified atmospheric model supporting various configurations, including severe weather nowcasting, hurricane forecasting, and subseasonal-to-seasonal prediction.
- Thanks to container technology SHiELD has become quite easy to use. Even better, SHiELD can do runs on all major operating systems (Windows, macOS, and Linux) as well as on different cloud systems.
- In this specific case, the container technology uses the flexibility of the model and therefore opens many possibilities for research and education.

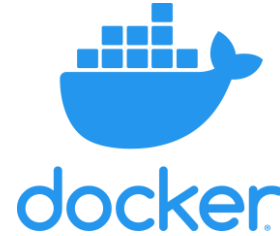
Example 2: GFDL: SHiELD v2021b in a container (2)

- For the purpose of demonstration and experimentation, 24-hour simulations of Hurricane Laura initialized from 2021082612 UTC were conducted.
- Two different domain configurations (regional and global-nested) were used to demonstrate the flexibility of SHiELD, showing that it can be used for a variety of applications at different spatial and temporal scales.

Machine	OS	CPU	Total cores
Laptop	Windows 10	Intel Core i7-8550U	4
Desktop	Windows 10	AMD Ryzen 9 3950X	16
Supercomputer	CentOS 7.6	Intel Xeon Gold 6148	72 000

Table 1: The operating System (OS), the central processing unit (CPU), and the total number of cores on the three representative machines. (Cheng et. al, 2021)

SHiELD in a Box: Setup



Docker (most popular container platform)

1. Install

```
docker pull gfdl fv3/shield
```

2. Download

Hurricane Laura case (regional model)
GFS fix files

3. Run

```
sudo docker run --cap-add=SYS_PTRACE -v  
PATH_of_CASE:/rundir -v  
PATH_of_GFS_fix:/GFS_fix -it gfdl fv3/shield  
mpirun -np 1 --allow-run-as-root  
SHiELD_nh.prod.32bit.x
```

Singularity (designed for supercomputer)

1. Install

```
singularity pull docker://gfdl fv3/shield
```

2. Download

Hurricane Laura case (global-nest model)
GFS fix files

3. Run

```
singularity exec --bind PATH_of_CASE:$HOME  
--bind PATH_of_GFS_fix:/GFS_fix  
PATH_of_image mpirun -np 1  
SHiELD_nh.prod.32bit.x
```

SHiELD in a Box: Results

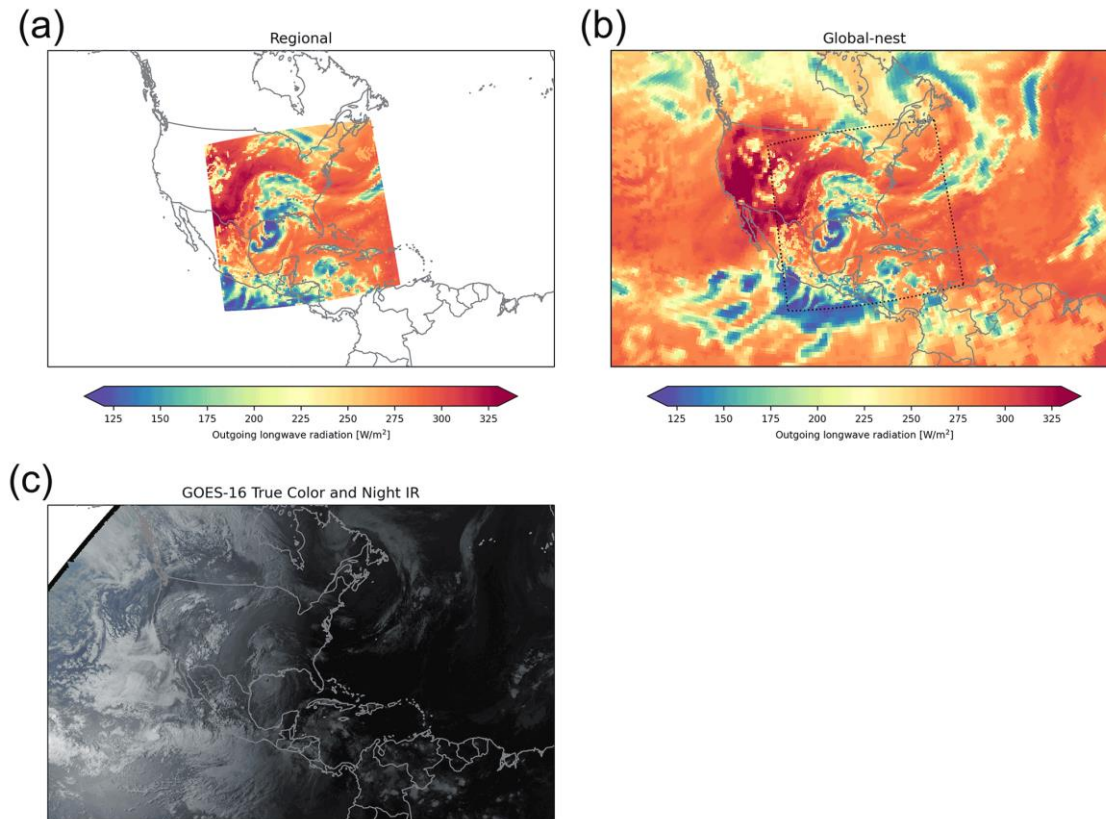


Figure 3: Hurricane Laura simulations versus satellite imagery. Panel (a) and (b) show outgoing longwave radiation simulated by SHiELD as a region and a global-nest model, respectively. Both are a snapshot at 00:00 UTC on 27th August 2020, 12 hours into the simulation. The dotted box represents the boundaries of the nest domain. Panel (c) shows the Geostationary Operational Environmental Satellite-16 true color and night infrared imagery 10 minutes after 00:00 UTC on 27th August 2020 (Cheng et. al, 2021)

SHiELD in a Box: Performance and Scalability

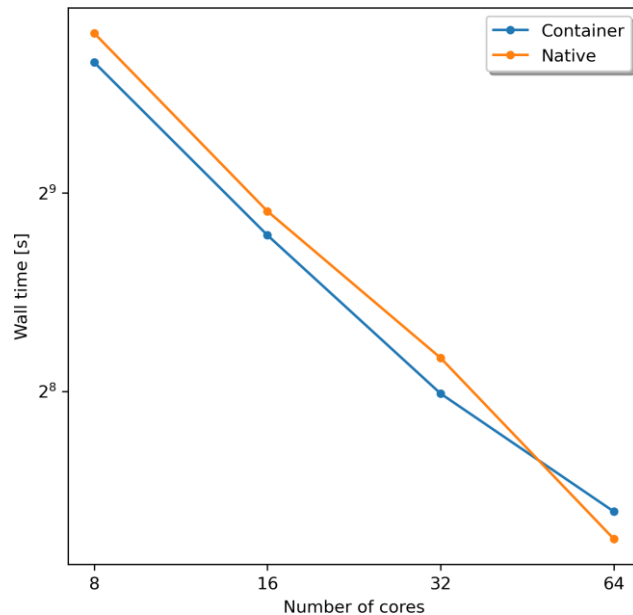


Figure 4: Performance and scalability of the containerized SHiELD (with Singularity), compared to those of the native SHiELD. Results are from 24h global-nest simulations. (Cheng et. al, 2021)

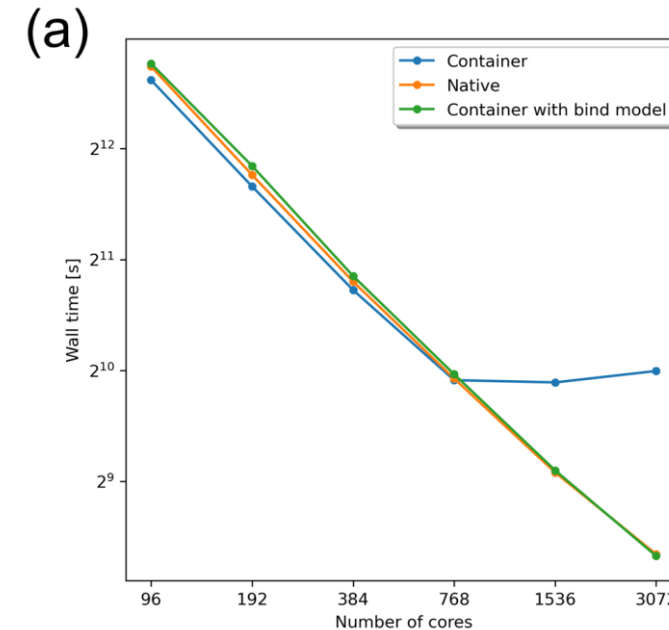


Figure 5: Performance and scalability of the containerized SHiELD, the native SHiELD, and the containerized SHiELD built with the bind model. Results are from 12 h global simulations at 13 km horizontal resolutions. (Cheng et. al, 2021)

Hand's On: Building a Basic Container

- To build a singularity container, one has to use the `build` command. It installs an OS, sets up the environment of the container and installs the application needed.
- In order to use the `build` command we need a definition file. This file is a set of instructions telling Singularity what software to install in the container. A simple definition looks like:

```
BootStrap: debootstrap
OSVersion: stable
MirrorURL: http://ftp.us.debian.org/debian/

%runscript
    echo "This is what happens when you run the container..."
%post
    echo "Hello from inside the container"
    apt-get -y --allow-unauthenticated install vim
%environment
    export PATH=$PATH:/usr/games
```

Hand's On: Developing a new Container

- Let's use the definition file to build our `lolcow.img` container. Note that the build command requires sudo privileges:

```
$ sudo singularity build --sandbox lolcow lolcow.def
```

- This is telling Singularity to build a container called `lolcow` from the `lolcow.def` definition file. The `-- sandbox` option tells Singularity that we want to build a special type of container for development purposes.
- Singularity can build containers in several different file formats (default is SIF), but if you want to „shell“ into a container, you should build a so called sandbox.
- When the build is finished, one will have a basic Debian container saved in a local directory called `lolcow`.

Hand's On: Explore the container & build SIF file

- In the next step we will enter the new container as root and look around:

```
$ sudo singularity shell --writable lolcow
```

- Now we try to install software inside:

```
Singularity> apt-get update
```

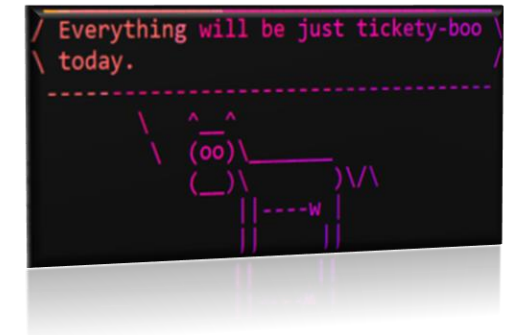
```
Singularity> apt-get install -y fortune cowsay lolcat
```

- Let's try running the demo in this new container:

```
Singularity> fortune | cowsay | lolcat
```

- Finally rebuild the container as SIF-Format:

```
$ sudo singularity build lolcow.sif lolcow.def
```



Outlook: Build a container for ICON (**ICONTAINER**)

- The goal of this project is to build a container for the climate model ICON-ESM using the already available source code.
- As a prerequisite, there should be an environment with stable modules that are essential for the use of ICON (e.g. `openmpi`).
- Once we have a File in the SIF-Format, one should simply can export this File on any HPC-System.
- In the future, one should submit a Slurm-Job like this:

```
#!/bin/bash
#SBATCH --job-name icontainer-test_run
#SBATCH -N 5 # total number of nodes
#SBATCH --time=00:10:00 # Max execution time
module load singularity
module load openmpi
mpirun -n $NP singularity exec icontainer.sif /icon-esm-runscripsts/parallel-test
```

References

- [1] K. Cheng, L. M. Harris, Y. Q. Sun (2021): Enhancing the accessibility of unified modeling systems: GFDL SHiELD v2021b in a container (<https://gmd.copernicus.org/articles/15/1097/2022/>)
- [2] F. Prill, C. Eser (2022): ICONIC – ICON in the Cloud
- [3] GFDL (2021): SHiELD in a Box (https://www.gfdl.noaa.gov/shield-2__trashed/shield-in-a-box/)
- [4] M. Blaschek (2021): User Workshop on Singularity in HPC
- [5] Sylabs Inc. (): Introduction to Singularity (<https://docs.sylabs.io/guides/3.5/user-guide/introduction.html>)
- [6] GitHub (2022): Singularity Tutorial (<https://singularity-tutorial.github.io/03-building/>)
- [7] IMGW (2021): Singularity (<https://gitlab.phaidra.org/imgw/singularity>)
- [8] ECMWF (2020): How to run OPENIFS in a Docker Container (<https://confluence.ecmwf.int/display/OIFS/How+to+run+OpenIFS+in+a+Docker+container>)