Motivation
Modern weather and climate models have massively parallel complex codes and complex workflow environments. Porting these codes is inherently difficult, as is installing, developing and maintaining the relevant infrastructure - particularly where reproducibility of workflow and output is required. The effort needed for this is a major drain on resources.

Model executables rely on a complex software stack to build, and associated workflow managers and tools require their own software stack. Ensuring scientific integrity of the modelling system when using different compilers and tools on different systems is another important consideration.

A container can help...

Singularity Container
This poster describes work containerising the ROSE/CYLC workflow management system and FCM build tools used by the Met Office’s Unified Model (UM) with a Singularity container. The container is built on a Linux system where the user has root permissions and includes the complete software stack required to build the model and the ROSE/CYLC workflow and FCM build tools. It is then copied to the target HPC.

Using the Container
ROSE/CYLC and FCM commands are run in scripts generated by the workflow manager and take usual command argument form eg: fcm build
These are replaced by a wrapper script to use the containerised version of the tool. The commands can then invoke further instances of themselves, but these will be run inside the container, so the wrapper takes the form of:

```bash
#!/bin/bash --login
if [-n "$SINGULARITY_CONTAINER" ]; then
    exec /container/bin/fcm "$@"
else
    exec singularity run -B /opt:/opt container.img fcm "$@"
fi
```

When the container is invoked on the HPC, the top level directories containing local software stack on the host HPC can be mounted as a bind (-B /opt:/opt above). Most HPCs now use modules, and as the module package is included in the container, it is then possible to access the HPC’s module configuration and software stack mounted under the bind. This permits use of local compilers and MPI libraries when building executables.

The model executable is built with the container, but is written to the user’s directory space, and is run outside the container environment using the local job control system, controlled and monitored by the external workflow manager.

Executable Portability
Using different compilers on different systems could lead to issues where models scientifically diverge. By careful use of compiler flags and static and shared libraries, it’s possible to build an executable suitable for a wide range of Intel based systems. Dependencies such as the Intel Fortran libraries and NetCDF inside the container can be compiled statically into the executable, negating the need for equivalent local libraries. If shared MPI libraries are used, then MPICH ABI allows the use of any compatible MPI, and therefore local fast interconnect, at run time.

Successes
The container has been used on a number of HPC services, a local cluster and a virtual cluster on running on a laptop, greatly reducing the effort of model installation. An executable built on a laptop has been run on a number of HPCs utilising MPICH ABI, including a Cray XC30. Progress is being made into making containerisation the standard method of all future model installations.

Issues
The process is reliant on consistent versions of Intel Fortran inside and outside the container. Environment variables set inside the container cannot be easily passed outside the container. Singularity is being actively developed, so can easily be outdated on HPCs. Requires root privileges to build the container.