Quick Start

The MusicBox code base can be found on GitHub. That web page gives instructions for running in a Docker Container.

Modifying the Environmental Conditions or Initial Conditions

Follow the usual process for development in a docker container:

```
git clone --recurse-submodules https://github.com/NCAR/MusicBox
cd MusicBox
docker build -t music-box-test . --build-arg TAG_ID=chapman
docker run -p 8000:8000 -it music-box-test bash
```

Install ftp in the container. Note that you are installing this in your container and not on your computer.

```
dnf -y install lftp
```

Collect the environmental conditions file. (These are time-evolving samples from a current-day simulation of CAM.)

```
cd /MusicBox/MusicBox_host/data/environmental_conditions
lftp ftp.acom.ucar.edu
lftp> cd micm_environmental_conditions
lftp> get Boulder.evolving_conditions.nc
lftp> exit
```

Note that ftp://ftp.acom.ucar.edu:/micm_environmental_conditions contains both environmental conditions and initial conditions for a several locations

- Boulder
- Seoul
- Global Data (Very large files (15Gb) that can take upwards of 10 minutes to download over a good connection)

Collect the initial condition.

```
cd /MusicBox/MusicBox_host/data/initial_conditions
lftp ftp.acom.ucar.edu
lftp> cd micm_environmental_conditions
lftp> ls
lftp> get Boulder.i.nc
lftp> exit
```

In /MusicBox/MusicBox_host, edit the file MusicBox_options to point to your new data:

```
 ! input file path
env_conds_file = '../data/environmental_conditions/Boulder.evolving_conditions.nc'
init_conds_file = '../data/initial_conditions/Boulder.i.nc'
```

Then build and run.

```
cd /MusicBox/MusicBox_host/src
make
```

```
./MusicBox
```

Running the code in a Docker container, if you have no viewing tool for netCDF files

For a Mac, you will have to open up xQuartz to allow x11 connections. This is a bit of a security hole.

- Open XQuartz
- set Preferences->Security->Allow connections from network clients to true
- restart XQuart

On other machines you may have to do similar operations.

Build and run model

From the terminal execute the following:

```
docker build -t music-box-test . --build-arg TAG_ID=chapman
xhost + 127.0.0.1
docker run -it -e DISPLAY=host.docker.internal:0 music-box-test bash
cd MusicBox/MusicBox_host/build
./MusicBox
ncview ../MusicBox_output.nc
```

Running another mechanism tag (chapman) in the same Docker container

From /MusicBox/Mechanism_collection directory, collect the tag from http://www.acom.ucar.edu/cafe, preprocess it to get the fortran for the solver, and stage it so that it can be complied.

```
./get_tag.py -tag_id chapman

./preprocess_tag.py -mechanism_source_path configured_tags/chapman -preprocessor localhost:3000

./stage_tag.py  -source_dir_kinetics configured_tags/chapman
```

From /MusicBox/MusicBox_host/build, compile the code and run it

```
make

./MusicBox
```

Running the code in a Unix environment

There are many dependencies in a UNIX environment. These are some of the dependencies that will need to be installed on your system.

- CMake3
- Fortran 2008 compiler
- git
- netcdf libraries
- wget
- python3
    - requests library
- node
    - express, helmet
- netcdf analysis and viewing tools
- Others that have been left off this list by accident

Environments including these dependencies have been set up on modeling2.acom.ucar.edu and cheyenne.ucar.edu

Steps for accessing and running the code

```
git clone https://github.com/NCAR/MusicBox
```

From the MusicBox directory, collect the rest of the code base

```
./manage_externals/checkout_externals
```

Get the environmental conditions for the box, and configure a tag (272) from the cafe-dev web server:

```
cd Mechanism_collection
python3 get_environmental_conditions.py
python3 get_tag.py -tag_id 272 -tag_server cafe-devel
python3 preprocess_tag.py -mechanism_source_path configured_tags/272
python3 stage_tag.py -soure_dir_kinetics configured_tags/272
```

Build the code

```
cd MusicBox_host
source etc/CENTOS_setup.sh -- or -- source etc/Cheyenne_setup_intel.sh

rm -rf build; mkdir build; cd build

cmake3 ../src/CMakeLists.txt -S ../src -B . -DCMAKE_BUILD_TYPE=Debug

make

Run the Code
/MusicBox
```

Users may find it informative to check out the options for any of the above python scripts

```
python3 burrito.py --help
python3 eat_it.py --help
python3 preprocess_tag.py --help
```

Adding parameterizations using CCPP

MusicBox uses the CCPP coding framework. This framework is meant to support "Plug-n-Play" for additional physical process parameterizations. Users may wish to contribute to the MusicBox effort.

Adding a new scheme

The steps to adding a new MusicBox scheme are as follows:
* Create XXX.F90 and XXX.meta files (the fortran code and its accompanying metadata file)
* Use "type = scheme" in the metadata file
* Add the XXX.meta file to MusicBox_scheme_files.txt
* Add the scheme to MusicBox_suite.xml in the order in which it is to be run

Adding a new ddt
The steps to adding a new ddt to MusicBox are as follows:
* Create XXX.F90 and XXX.meta files (a fortran module that contains the ddt and its accompanying metadata file)
* The metdata file will use "name = Name_of_your_ddt" and "type = ddt".
* Each element in the ddt will be documented with its local name, standard name, type, etc.
* Add the XXX.meta file to MusicBox_ddt_files.txt
* Add the ddt to MusicBox_mod.F90 and MusicBox_mod.meta
* The ddt can then be passed in/out and "use"d within any module