Questions?

Reproducible Development with Scripts

What is Reproducible Development?

- NorESM is a highly configurable model
  - Model tag (need to check out NorESM tag and update all externals)
  - Component set (set of active and data component models)
  - Model configuration options for each active or data component
  - Runtime (namelist) options for each component (including output options)
  - These configuration steps must be made at the correct time in the experiment creation.
  - The easiest way to do this in a manner that can be reproduced is to put them in a (Bash) script

Reproducible Development with Scripts

- Most aspects of reproducible experiments can be included in a simple bash script
- A script is a single file that can be shared and stored.
  - In addition to creating all the settings found in the case documentation, it can (should) also control the exact code version used for the experiment.
- Elements of a script
  1) Make sure the model source (`SRCROOT`) has the correct version of the model (`TAG`) checked out (including infrastructure and component models)
  2) Call **`create_newcase`** with a compset (`COMPSET`), model grid (`RES`), case (`CASEDIR`) and any other desired options
  3) Execute any need **`xmlchange`** commands
  4) Call **`case.setup`**
  5) Add any desired custom namelist settings
  6) Call **`case.build`** and **`case.submit`**
  7) Check for errors in any of these steps

Sample Experiment Script (part 1)

```
#! /bin/bash
## Run an out-of-the-box N2000 experiment for 3 months
##    but with reduced output.

## Experiment basics, modify these for your experiment
TAG="noresm2_3_beta01"
COMPSET="N2000"
RES="f19_tn14"
SRCROOT="/cluster/projects/nn9560k/xxUSERxx/NorESM"
CASEDIR="/cluster/work/users/xxUSERxx/cases/${COMPSET}_${RES}"
REPO="https://github.com/NorESMhub/NorESM"
PROJECT="nn9560k"

## (make sure that clone exists, otherwise, clone REPO)
if [ ! -d "${SRCROOT}" ]; then
    git clone -o NorESM ${REPO} ${SRCROOT}
fi

## Ensure correct source is checked out
cd ${SRCROOT}
git checkout ${TAG}
./manage_externals/checkout_externals
```

Sample Experiment Script (part 2)

```
## Create your case
./cime/scripts/create_newcase --case ${CASEDIR} --compset ${COMPSET} --res $
{RES}

## Any PE changes must go here

## Setup the case
./case.setup

## Changes that affect the build go here
./xmlchange DEBUG=TRUE
./xmlchange STOP_OPTION=nmonths,STOP_N=3

## Build the model
./case.build

## Last chance to modify run-time settings
echo "history_chemistry       = .false." >> user_nl_cam
echo "history_chemspecies_srf = .false." >> user_nl_cam
echo "history_clubb           = .false." >> user_nl_cam

## Submit the job
./case.submit
```

Sample Scripts on Betzy

There are two sample scripts on Betzy:

- **`/cluster/shared/noresm/WORKSHOP/scripts/ReproExperimentScriptSimple.sh`**
  - Just run the commands.
  - Is nearly the same as the example on the previous two pages
- **`/cluster/shared/noresm/WORKSHOP/scripts/ReproExperimentScript.sh`**
  - A more careful script that performs checks for success and also can skip some steps if they have already been done

To work with one of these scripts, make a copy and then make changes to suite your needs. In particular, review these four lines and at least replace xxUSERxx with your Betzy user ID.

```
COMPSET="N2000"

RES="f19_tn14"

SRCROOT="/cluster/projects/nn9560k/xxUSERxx/NorESM"

CASEDIR="/cluster/work/users/xxUSERxx/cases/${COMPSET}_${RES}"
```

Questions?

More on namelist variables

- A namelist is a Fortran feature that makes it easy to read (and write) data. It allows you to directly set many variables in the code at runtime with a single read command.
- Most NorESM model components use namelists to configure their runtime environments.
- Models such as CAM (the atmosphere component) are highly configurable; CAM has over 1000 namelist variables.
- Most namelist variables will have default values so you only have to set ones where you want to use a different value.
- The CIME case infrastructure provides you a way to set only the namelist values you need to change. Each model component has a file for this:
  - user_nl_blom (ocean component) user_nl_cam (atmosphere component)
  - user_nl_cice (sea ice component) user_nl_cism (land ice component
  - user_nl_clm (land component) user_nl_cpl (settings for the driver and mediator)
  - user_nl_mosart (river runoff component)

More on namelist variables (2)

- The user_nl_<component> files are created during the **./case.setup** step.
- The values you add to the user_nl_<component> files are folded into the full component namelist files before your experiment is submitted to the queue.
- You can also do this operation manually using the **./preview_namelists** command in your case directory.
- The full namelist files are in your run directory but also copied to the CaseDocs folder in your case directory. For instance:
  - user_nl_cam ==> atm_in
  - user_nl_blom ==> ocn_in
  - user_nl_clm ==> lnd_in
- These files are automatically generated, any edits will be lost!

More on namelist variables (3)

- What does a Fortran namelist look like?

```
&cldfrc_nl ! Name of the namelist
  cldfrc_dp1 = 0.10D0
  cldfrc_dp2 = 500.0D0
  cldfrc_freeze_dry = .true.
  cldfrc_ice = .true.
  cldfrc_icecrit = 0.93D0
/
```

- If you want to change a variable, just add it with the new value to your user_nl_<component> file.

```
  cldfrc_dp1 = 0.05D0
```

Questions?