# CICE Consortium tutorial - Standalone CICE and Icepack activity

## Learning Goals

In this activity you will clone the CICE and Icepack model code from the Consortium GitHub repository to the Cheyenne supercomputer and run standalone CICE and Icepack simulations. You will also make namelist changes and code modifications for experiments and compare these to control simulations.

Notes:

- A line with a right-facing angle bracket (>) followed by text, denotes a command you need to enter at the command line on the supercomputer but without the angle bracket.
- When there is the <X> syntax, you need to fill in your personal information (e.g. a URL or username) for that command but without the angle brackets. Your GitHub and Cheyenne usernames may not be the same so check which you need to use.
- ***You may work entirely on your own for these activities, but we recommend you find a partner or small group to turn to for help, discuss the questions, and coordinate some of the afternoon experiments.***

## Part 0 - GitHub

Please see the document sent before the tutorial about starting with GitHub. **You need to have your own GitHub account before you can start the following activities and you should have forked the Consortium repositories already.**

Double check that your fork is up to date with the most recent Consortium master branches. If you need to know how to do this, look at the comments under #2 of “Creating your own Forks”.

## Part 1 - Getting Started (~1hr)

This morning you will be starting to work with Icepack and CICE in their standalone form. You will get the model code from GitHub and run out-of-the-box (all default options) experiments using the latest release of the Icepack and CICE code.

### *Starting on the Cheyenne Supercomputer*

1. Log into Cheyenne

You will run all the activities for this workshop on Cheyenne, so this step will be repeated in future activities and you can refer here for access.

Start ssh connection through terminal window:

```
>ssh -Y <username>@cheyenne.ucar.edu
```

Note: on your laptop you need both the -Y and -X arguments here to visualize graphics.

Your screen will display the following response: Token_Response:

How you authenticate can be different depending if you have a yubikey or use Duo.

a) Access with yubikey:
   - Plug yubikey into USB port on your computer.

  - Enter PIN number, but **do not hit enter**. Lightly touch the yubikey button which will automatically complete and submit your password.

b) Access with Duo:
  - Enter CIT password
  - Duo will push an authorization request to your phone. After you confirm the request you will be logged in.

2. Set up Cheyenne for this tutorial

After you log in to Cheyenne, you will be in your home directory (/glade/u/home/<username>). You can run the CICE and Icepack code from here, but you should not store any big files here as the space quota is small.

In your home directory create and the move to the following directory:

```
>mkdir cice-tutorial
>cd cice-tutorial
```

## *Run out-of-the-box Icepack*

1. Clone **your** Icepack repository fork (use the URL from your fork) to a local sandbox.

```
>git clone <URL to repository> icepack_master
```

If you have completed this correctly there should be a “icepack_master” directory in the cice-tutorial directory. This is the “sandbox” we will be working in locally on Cheyenne.

2. Move to the icepack_master directory on Cheyenne and check which branch you are using.

```
>cd ~/cice-tutorial/icepack_master
>git status
```

Take a minute to orient yourself to the big picture structure of the directories and files in Icepack. The documentation has information about the Icepack Directory Structure (section 3.1.1).

3. Set up an out-of-the-box Icepack simulation.

Use the online Icepack documentation and in particular the “Quick Start” (section 1.2) and “Running Icepack” (section 3.2) parts as guidance and for details on what each command line option sets.

```
>./icepack.setup --case ~/cice-tutorial/cases/icepack_test0 --mach cheyenne --env intel -
-acct UCGD0007 --queue S623186
```

The setup script creates a case consistent with the machine and other defined settings under ~/cice-tutorial/cases/ with the name you selected (icepack_test0). The case directory will contain build and run scripts, a namelist file, and other necessary files. Once the case is set up any of these files can be manually edited to refine the desired configuration.

Note that the account key and queue you are using are specific to this tutorial. If you run this elsewhere or on Cheyenne outside this tutorial you will need a different account key and queue (you can use the share queue on Cheyenne).

4. Move to the new case directory and examine the settings.

```
>cd ~/cice-tutorial/cases/icepack_test0
```

Open the icepack.settings file and look at it briefly. Because Cheyenne is a supported machine, Icepack already "knows" specific machine settings. Note the ICE_CASEDIR (it should match this directory) and the ICE_RUNDIR (where the model will be run and output created).

Now look at the default namelist settings in icepack_in.

5. Build the code.

```
>qcmd -A UCGD0007 -- ./icepack.build
```

The build script basically runs gmake under the covers, but there are a number of other tasks that are handled by the script to make the build more robust.

<u>The qcmd command is specific to the Cheyenne system</u>; it creates a job to build the model and submits it on the fly and is designed to prevent the login nodes from being overwhelmed. qcmd needs the -A argument to specify which account to charge for the build. Normally you would just type ./icepack.build on the command line.

If the build is successful you will see the message "COMPILE SUCCESSFUL" at the bottom of the screen. You can also check the README.case file to check the status.

6. Submit the job. The submit script just submits the run scripts. Look at both icepack.run and icepack.submit to see more details.

```
>./icepack.submit
```

Some useful commands on Cheyenne:
To check the status of a job or get a jobid: >qstat -u <username>
To kill a job: >qdel <jobid>

If the run is successful, you will see the message "ICEPACK COMPLETED SUCCESSFULLY" in the icepack_test0.o* file. Note that this job runs quickly - you are just running a single column!

7. Look at the output!

Go to the ICE_RUNDIR where output was created. A successful model integration will create ice_diag.* files and a file in the "restart" directory called "iced.2016-01-01-00000". The Icepack documentation has more information about "Model output" (section 3.1.5).

Follow the documentation to create some plots of the output using the tools provided with Icepack (see section 3.3.4).

```
>cd ~/cice-tutorial/icepack_master/configuration/scripts/tests/
>./timeseries.csh
/glade/scratch/<username>/ICEPACK_RUNS/icepack_test0/ice_diag.full_ITD
```

Note that you can run the plotting script on any of the four ice_diag.* files.

The *.png files are created in the ICE_RUNDIR directory. Open the files:

```
>cd /glade/scratch/<username>/ICEPACK_RUNS/icepack_test0/
>eog <figurename>.png
(Note: the eog display option will allow you to scroll through multiple images)
# or
>display <figurename>.png
#or
You can copy the files back to your laptop to view
```

8. Questions to discuss while looking at the output.

- What time period does an out-of-the-box run cover?
- What are the differences between the full_ITD plots and the icefree plots (or any other combination of the ice_diag.* output files)? Which fields are the same? Which are different? Why would this be?
- What happens to ice area and ice thickness around October 1, 2015? Why do you see this signal?
- How does your output compare to the sample output provided for this release? (hint, see the wiki!)

9. Big Picture of the Directory Structure. Take a step back and think about all the directories and files you have created. This workflow will be similar for CICE as well.

   a) The icepack_master “sandbox” was cloned from GitHub and has the actual Icepack code.
   b) There is a particular case directory for building and launching the code, and some output (e.g. job log) are copied.
   c) There is a particular run directory for each case. This is where the model is run and big files are found.

## *Run out-of-the-box CICE*

1. Return to the main directory for this tutorial

```
>cd ~/cice-tutorial/
```

2. Clone **your** CICE repository fork (use the URL from your fork) to a local sandbox.

Additionally, Icepack is a submodule of CICE, so to also download Icepack within CICE add the “--recursive” argument when cloning the CICE code.

```
>git clone <URL to repository> cice_master --recursive
```

If you have completed this correctly there should be a new “cice_master” directory.

3. Move to the cice_master directory on Cheyenne and check which branch you are using.

```
>cd ~/cice-tutorial/cice_master
>git status
```

Take a minute to orient yourself to the big picture structure of the directories and files in CICE.

There is an “icepack” directory that should have files and code that match the structure in the Icepack directory. If you did not clone with the --recursive flag, the “icepack” directory will be empty. More information about submodules, including how to check what version of Icepack is being pointed at, can be found in the Consortium Workflow guidance documentation: https://github.com/CICE-Consortium/About-Us/wiki/Git-Workflow-Guidance#submodules

4. Set up an out-of-the-box CICE simulation.

```
>./cice.setup --case ~/cice-tutorial/cases/cice_test0 --grid gx3 --mach cheyenne --env
intel --set diag1 --pes 8x1 --acct UCGD0007 --queue S623186
```

This will automatically create a directory under ~/cice-tutorial/cases/ with the name you selected (cice_test0). CICE requires more arguments than Icepack to set up a case. Use the CICE documentation and in particular the “Quick Start” (section 1.2) and “Running CICE” (section 3.2) parts as guidance and for details on what each command line option sets. You have used the long-form version of each option here, but there are also short-form versions for many options that we will use this afternoon.

5. Move to the new case directory and examine settings.
```
>cd ~/cice-tutorial/cases/cice_test0
```

Open the cice.settings file and look at it briefly. As with Icepack, Cheyenne is a supported machine so case should build and run without any problems.

Now look at the default namelist settings in the ice_in file.

6. Build the CICE code.
```
>qcmd -A UCGD0007 -- ./cice.build
```

Again, normally you would just type ./cice.build at the command line to build CICE. If the build is successful you will see the message "COMPILE SUCCESSFUL" at the bottom of the screen. You can also check the README.case file to check the status.

7. Submit the CICE job.
```
>./cice.submit
```

If the run is successful, you will see the message "CICE COMPLETED SUCCESSFULLY" in the cice_test0.o* file. Note that this job runs quickly too.

8. Watch the queue and when the job is complete, look at the output!

Go to the run directory, where output was created (see cice.settings for ICE_RUNDIR). A successful model integration will create a "history" and "restart" directory that both have NetCDF files. There is information about CICE output in the online documentation (see section 3.1.5).

Look at these output files:
```
>module load ncview
>ncview <filename>.nc
```

9. Questions to think about while looking at the output (the namelist may help):

- What are the thermodynamics and dynamics timesteps?
- For how long did this test integrate? Why was the runtime so fast?
- How would you change the duration of the run?
- When would a restart run begin?
- What is the spatial domain of CICE? What type of grid does CICE use?
- How does this compare to the sample output provided for this release?

## *Run CICE longer*

Once you have had success with the previous step, you should run another out-of-the-box for a longer experiment in order to practice some basic changes for CICE.

1. Go back to your cice_master directory
```
>cd ~/cice-tutorial/cice_master/
```

2. You need to set up a new out-of-the-box case (cice_test1), but with a different option in cice.setup. What is the change below? What are the other options for this value?

```
>./cice.setup --case ~/cice-tutorial/cases/cice_test1 --grid gx3 --mach cheyenne --env
intel --set diag1,short --pes 8x1 --acct UCGD0007 --queue S623186
```

3. Identify what's changed in the cases.

   ```
   >cd ~/cice-tutorial/cases/
   >diff -r cice_test0 cice_test1
   ```

4. Go into the cice_test1 directory, and build the case.

5. Change the following namelist settings in ice_in:

   npt = 2160 - How long is this setting the model to run?
   histfreq = 'm','d','h','x','x' - What does this change?
   histfreq_n     =  1 , 1 , 6 , 1 , 1 - what will happen here?
   f_aice = 'md' - What is this variable? How often will it be output?
   f_hi = 'md' - What is this variable? How often will it be output?
   f_uvel = 'h' - What is this variable? How often will it be output?
   f_vvel = 'h' - What is this variable? How often will it be output?
   f_uatm = 'x' - What is this variable? How often will it be output?
   f_vatm = 'x' - What is this variable? How often will it be output?

   Details about namelist options are in the documentation (section 3.4.2)

6. Submit the job. Check the output and think about the following:
   - Over what dates did the model run this time?
   - When would the model restart from?
   - What is in each type of file and how does the output naming differ for the files?
   - Does the history output match what you expected given the ice_in changes you made?

7. Now you want to restart the model with a few additional changes.
   - Change all the variables except f_aice and f_hi back to output monthly fields only.
   - runtype = 'continue'
   - Note, but do not change, the restart_dir and pointer_file values.

8. Submit the job. Does the output match what you expect? Is there anything unexpected?

9. Go to the seminar, have lunch, go for a walk. Recharge for the afternoon session!