Porting

Tony Craig, Philippe Blain

CICE and Icepack Workshop and Tutorial February 3-5, 2020

Porting Overview

- Get the model running on a new machine
- For the standalone models
- Documented in the user guide
- This is where the machine and env names are defined for use in the model scripts
- System Requirements
    - Icepack and CICE require
        - Fortran and C compiler
        - GNU make for compilation
    - CICE works better if you can use
        - MPI
        - NetCDF
- ToDo:
    - Download input datasets and place in a non- scrubbed directory (see GitHub documentation)
    - Port model to machine
    - PR porting modifications to the Consortium Repo to support reuse

Scripts Porting

- Four primary issues to address
  1. Environment setup (compiler version, system libraries, machine settings, etc)
  2. Compilation commands and flags
  3. Batch syntax and submission
  4. Binary launching

1. Environment Setup

- Defines the compiler version and other system libraries (often using system modules)
- Defines some machine settings (number of processors per node, default baseline run directory, input data directory, etc)
- Setting defined in a file named
  - *configuration/scripts/machines/env.$mach_$env*
  - where $mach and $env are the machine and environment name chosen. These will be used with –mach and –env in the model scripts.
- ToDo: Copy an existing file and modify
- See documentation for detailed information about env variables

env.cheyenne_intel

```
#!/bin/csh –f

set inp = "undefined”
if ($#argv == 1) then
  set inp = $1
endif

if ("$inp" != "-nomodules") then

source /glade/u/apps/ch/opt/lmod/7.2.1/lmod/7.2.1/init/csh
module purge
module load ncarenv/1.2
module load intel/19.0.2
module load mpt/2.19
module load ncarcompilers/0.5.0
module load netcdf/4.6.3

endif

setenv ICE_MACHINE_ENVNAME cheyenne
setenv ICE_MACHINE_COMPILER intel
setenv ICE_MACHINE_MAKE gmake
setenv ICE_MACHINE_WKDIR /glade/scratch/$user/CICE_RUNS
setenv ICE_MACHINE_INPUTDATA /glade/p/cesm/pcwg_dev
setenv ICE_MACHINE_BASELINE /glade/scratch/$user/CICE_BASELINE
setenv ICE_MACHINE_SUBMIT "qsub”
setenv ICE_MACHINE_ACCT P00000000
setenv ICE_MACHINE_QUEUE "regular”
setenv ICE_MACHINE_TPNODE 36
setenv ICE_MACHINE_BLDTHRDS 1
setenv ICE_MACHINE_QSTAT "qstat "
```

2. Compilation

- Compile by running the *$case.build* script which uses gmake under the covers
- A generic Makefile is provided with Icepack and CICE, *configuration/scripts/Makefile*
- Machine specific settings are defined in a Macros file
  - *configuration/scripts/machines/Macros.$mach_$env*
  - Use the same $mach_$env string as defined with the env file.
- ToDo: Copy an existing file and modify

Macros.cheyenne_intel

```
#===============================================================================
# Makefile macros for NCAR cheyenne, intel compiler
#===============================================================================

CPP        := fpp
CPPDEFS    := -DFORTRANUNDERSCORE ${ICE_CPPDEFS}
CFLAGS     := -c -O2 -fp-model precise   -xHost

FIXEDFLAGS := -132
FREEFLAGS  := -FR
FFLAGS     := -fp-model precise -convert big_endian -assume byterecl -ftz -traceback   -xHost
FFLAGS_NOOPT:= -O0

ifeq ($(ICE_BLDDEBUG), true)
  FFLAGS     += -O0 -g -check uninit -check bounds -check pointers -fpe0 -check noarg_temp_created
else
  FFLAGS     += -O2
endif

SCC   := icc
SFC   := ifort
MPICC := mpicc
MPIFC := mpif90
i
feq ($(ICE_COMMDIR), mpi)
  FC := $(MPIFC)
  CC := $(MPICC)
else
  FC := $(SFC)
  CC := $(SCC)
endif
LD:= $(FC)

NETCDF_PATH := $(NETCDF)

PIO_CONFIG_OPTS:= --enable-filesystem-hints=gpfs

#PNETCDF_PATH := $(PNETCDF)
#PNETCDF_PATH := /glade/u/apps/ch/opt/pio/2.2/mpt/2.15f/intel/17.0.1/lib

INCLDIR := $(INCLDIR)

LIB_NETCDF := $(NETCDF_PATH)/lib
#LIB_PNETCDF := $(PNETCDF_PATH)/lib
LIB_MPI := $(IMPILIBDIR)

#SLIBS   := -L$(LIB_NETCDF) -lnetcdf -lnetcdff -L$(LIB_PNETCDF) -lpnetcdf –lgptl
SLIBS   := -L$(LIB_NETCDF) -lnetcdf –lnetcdff
i
feq ($(ICE_THREADED), true)
  LDFLAGS += -qopenmp
  CFLAGS += -qopenmp
  FFLAGS += -qopenmp
endif

### if using parallel I/O, load all 3 libraries.  PIO must be first!
ifeq ($(ICE_IOTYPE), pio)
  PIO_PATH:=/glade/u/apps/ch/opt/pio/2.2/mpt/2.15f/intel/17.0.1/lib
  INCLDIR += -I/glade/u/apps/ch/opt/pio/2.2/mpt/2.15f/intel/17.0.1/include
  SLIBS   := $(SLIBS) -L$(PIO_PATH) –lpiof
endif
```

3. Batch support

- The batch scripts allow the model to be submitted to a machine in batch mode. These scripts are added to the top of the *$case.run* and *$case.test* scripts
- There are a handful of batch applications (PBS, slurm, etc), but each install is typically customized based on particular machine requirements
- The batch scripts are defined in
    - *configuration/scripts/cice.batch.csh*
- ToDo: Add an if block to that file for your machine
    - ICE_MACHINE will be *$mach_$env* in the resolved scripts
    - Batch settings usually do not depend on compiler
    - Leverage the node, processor, account, queue, and run length computations at the top of the script. Add other computations if needed.

cice.batch.csh

```
...

set ntasks = ${ICE_NTASKS}
set nthrds = ${ICE_NTHRDS}
set maxtpn = ${ICE_MACHINE_TPNODE}
set acct   = ${ICE_ACCOUNT}

@ ncores = ${ntasks} * ${nthrds}
@ taskpernode = ${maxtpn} / $nthrds
if (${taskpernode} == 0) set taskpernode = 1
@ nnodes = ${ntasks} / ${taskpernode}
if (${nnodes} * ${taskpernode} < ${ntasks}) @ nnodes = $nnodes + 1
set taskpernodelimit = ${taskpernode}
if (${taskpernodelimit} > ${ntasks}) set taskpernodelimit = ${ntasks}
@ corespernode = ${taskpernodelimit} * ${nthrds}
...
set queue = "${ICE_QUEUE}”
set batchtime = "00:15:00"
...

if (${ICE_MACHINE} =~ cheyenne*) then
cat >> ${jobfile} << EOFB
#PBS -j oe
###PBS -m ae
#PBS –V
#PBS -q ${queue}
#PBS -N ${ICE_CASENAME}
#PBS -A ${acct}
#PBS -l select=${nnodes}:ncpus=${corespernode}: \
    mpiprocs=${taskpernodelimit}:ompthreads=${nthrds}
#PBS -l walltime=${batchtime}
EOFB

...
```

4. Binary launch

- Job launching is done in the *$case.run* script
- The binary executable launch is often a function of the batch system, MPI library, and/or machine. Note that Icepack launches by default as a serial application (./icepack), but that is not universal.
- The binary launch is defined in
  - *configurations/scripts/cice.launch.csh*
- ToDo: Add an if block to that file for your machine
  - ICE_MACHINE will be *$mach_$env* in the resolved scripts
  - Launch settings usually do not depend on compiler
  - Leverage the task and thread computations at the top of the script. Add other computations if needed.

cice.launch.csh

```
...

set ntasks = ${ICE_NTASKS}
set nthrds = ${ICE_NTHRDS}
set maxtpn = ${ICE_MACHINE_TPNODE}

@ ncores = ${ntasks} * ${nthrds}
@ taskpernode = ${maxtpn} / $nthrds
if (${taskpernode} == 0) set taskpernode = 1
@ nnodes = ${ntasks} / ${taskpernode}
if (${nnodes} * ${taskpernode} < ${ntasks}) @ nnodes = $nnodes + 1
set taskpernodelimit = ${taskpernode}
if (${taskpernodelimit} > ${ntasks}) set taskpernodelimit = ${ntasks}
@ corespernode = ${taskpernodelimit} * ${nthrds}

...

if (${ICE_MACHINE} =~ cheyenne*) then
if (${ICE_COMMDIR} =~ serial*) then
cat >> ${jobfile} << EOFR
./cice >&! \$ICE_RUNLOG_FILE
EOFR
else
cat >> ${jobfile} << EOFR
mpiexec_mpt -np ${ntasks} omplace ./cice >&! \$ICE_RUNLOG_FILE
EOFR
endif

...
```

Porting: Validation

- Create a case using the new machine and env values
- Test build and run scripts. Refine as needed in the case
  - env and Macros files can be copied back to the configuration/scripts/machines directory
  - Batch and launch modifications will need to be manually updated in the batch and launch scripts in configuration/scripts
- Create a new case and continue to refine until working
- Run a test suite to technically validate
- Run a qc test versus a known baseline to validate science and/or carry out a longer run and review results

Porting Summary

- Download the input data and place in a non-scrubbed space
- Establish a machine and env name
- Add a *configuration/scripts/machines/env.$mach_$env* file
- Add a *configuration/scripts/machines/Macros.$mach_$env* file
- Add a block of code to *configuration/scripts/cice.batch.csh*
- Add a block of code to *configuration/scripts/cice.launch.csh*
- Test, refine, and validate
- PR back to the Consortium repo (optional but recommended)
- See documentation for additional details. Leverage the forum to see answered questions and to ask new ones

Porting to a personal computer

- Get CICE running quickly on your personal machine
- Less involved than the full porting process
- Useful for model development
- Leverages the conda package manager

CONDA

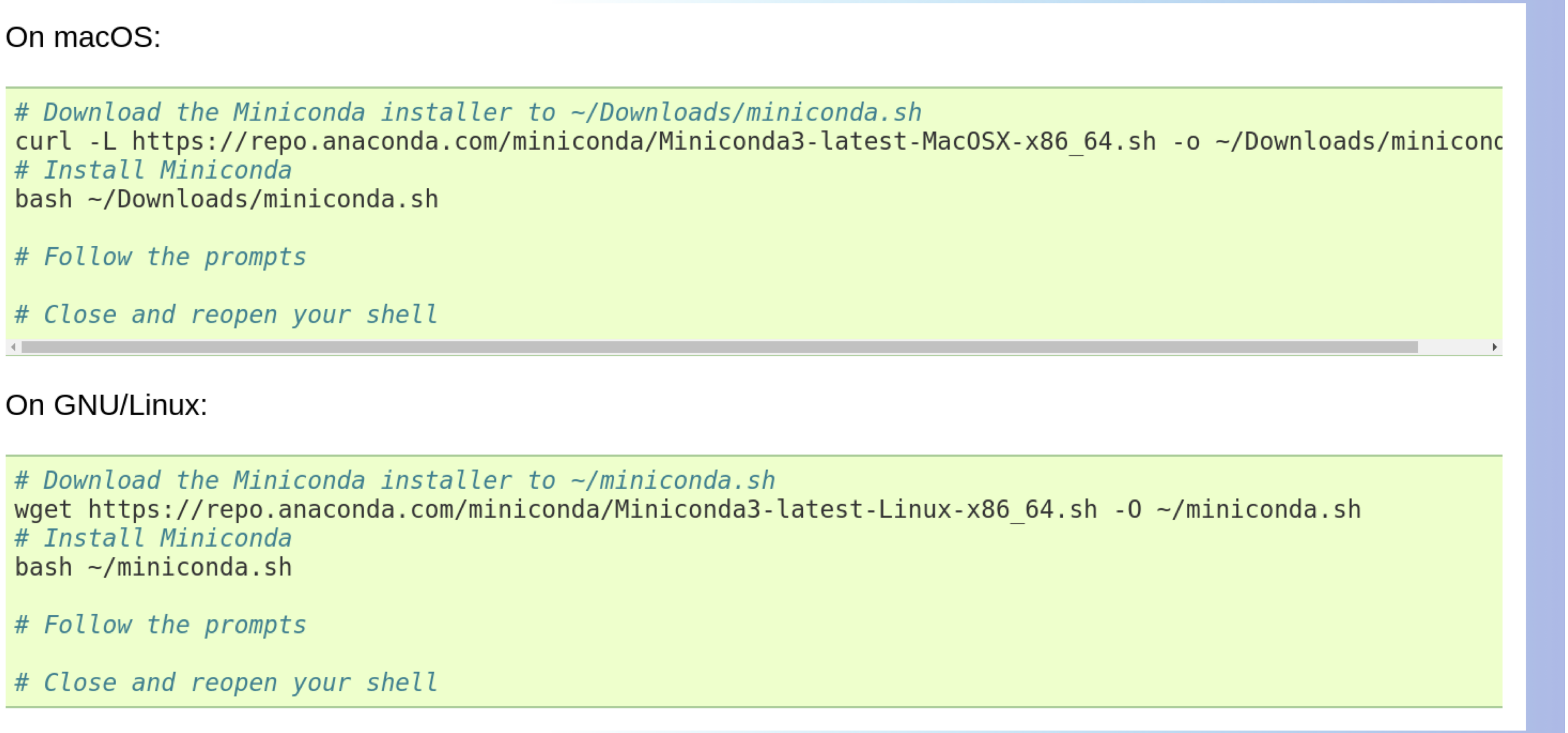
Steps:

1. Install Miniconda
2. Create the “cice” conda environment
3. Run CICE !

CICE Consortium

1. Installing Miniconda

Download installer script and follow instructions:



On macOS:

```
# Download the Miniconda installer to ~/Downloads/miniconda.sh
curl -L https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-x86_64.sh -o ~/Downloads/minicond
# Install Miniconda
bash ~/Downloads/miniconda.sh

# Follow the prompts

# Close and reopen your shell
```

On GNU/Linux:

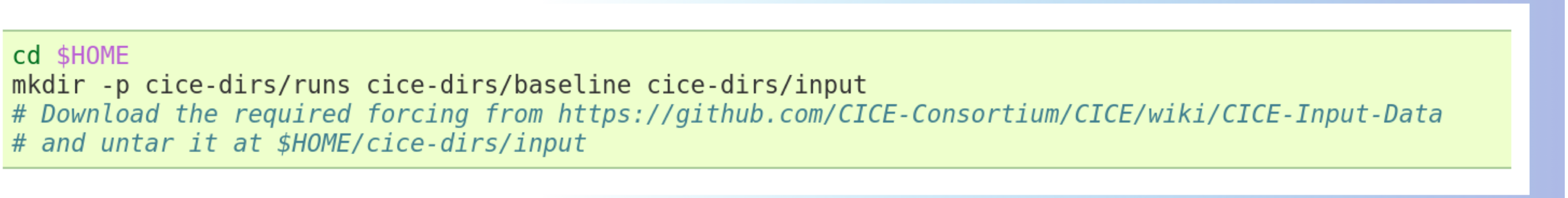
```
# Download the Miniconda installer to ~/miniconda.sh
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O ~/miniconda.sh
# Install Miniconda
bash ~/miniconda.sh

# Follow the prompts

# Close and reopen your shell
```

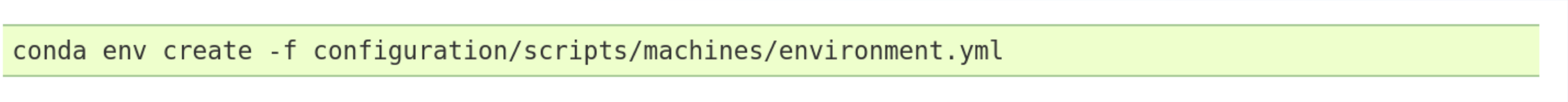
2. Creating the environment

- Create required folders in your `$HOME`



```
cd $HOME
mkdir -p cice-dirs/runs cice-dirs/baseline cice-dirs/input
# Download the required forcing from https://github.com/CICE-Consortium/CICE/wiki/CICE-Input-Data
# and untar it at $HOME/cice-dirs/input
```

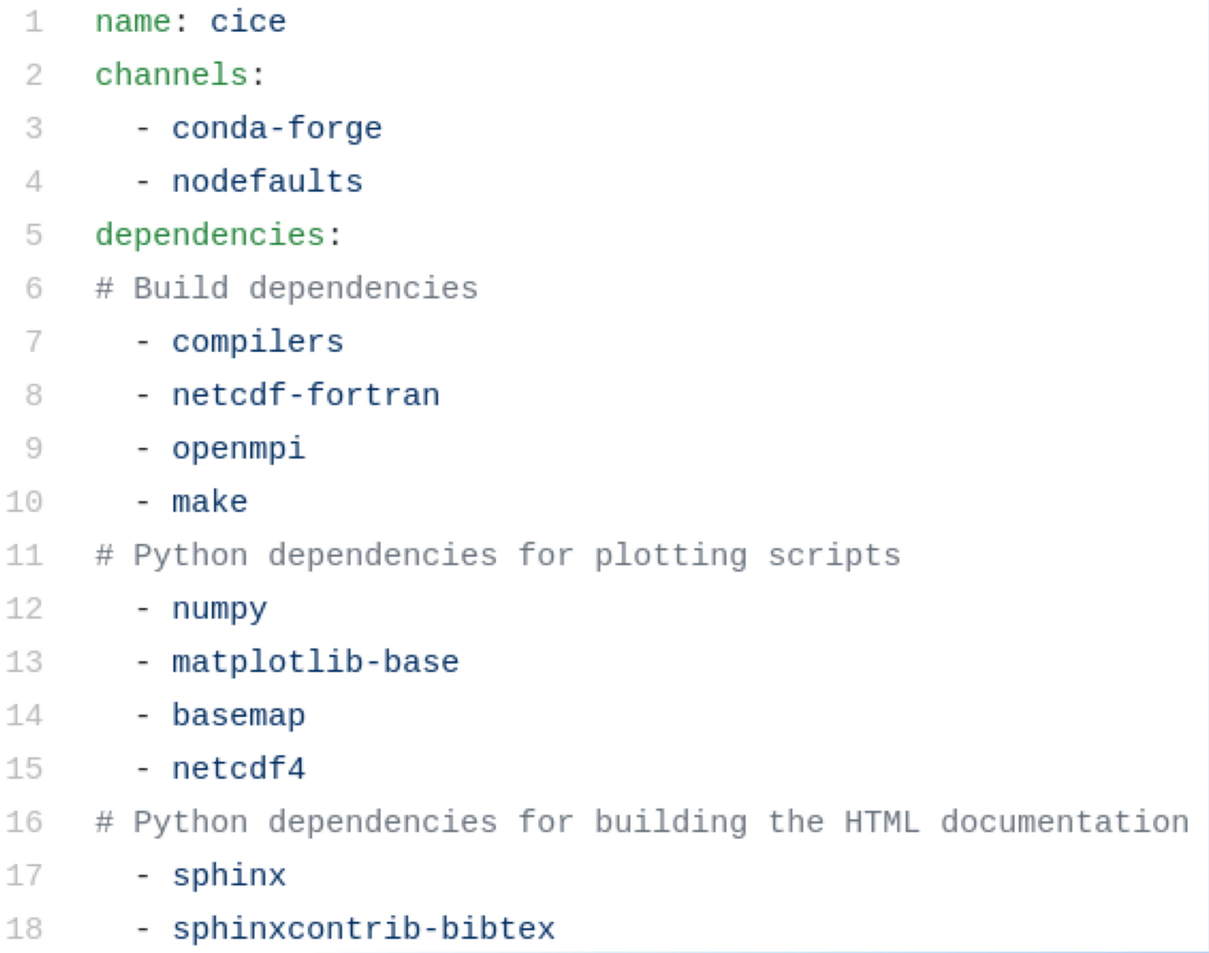
- Download and untar forcing to `$HOME/cice-dirs/input`
- Create the conda environment



```
conda env create -f configuration/scripts/machines/environment.yml
```

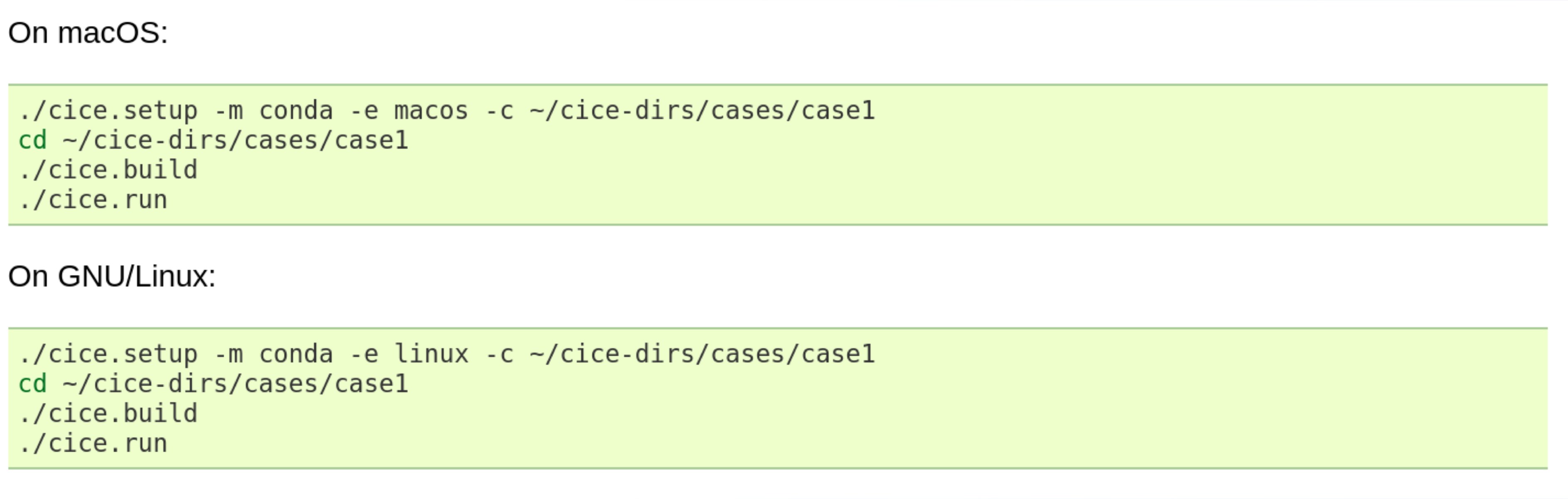
2. Creating the environment

./configuration/scripts/machines/environment.yml



```
name: cice
channels:
  - conda-forge
  - nodefaults
dependencies:
# Build dependencies
  - compilers
  - netcdf-fortran
  - openmpi
  - make
# Python dependencies for plotting scripts
  - numpy
  - matplotlib-base
  - basemap
  - netcdf4
# Python dependencies for building the HTML documentation
  - sphinx
  - sphinxcontrib-bibtex
```

3. Using the environment to run CICE



On macOS:

```
./cice.setup -m conda -e macos -c ~/cice-dirs/cases/case1
cd ~/cice-dirs/cases/case1
./cice.build
./cice.run
```

On GNU/Linux:

```
./cice.setup -m conda -e linux -c ~/cice-dirs/cases/case1
cd ~/cice-dirs/cases/case1
./cice.build
./cice.run
```

- No batch system (interactive only)
- Don’t run big test suites!
- Use `conda activate cice` for plotting scripts and building documentation
- More details in documentation