

Hi everyone! I'm Brian Dobbins, and I'm a software engineer here at NCAR. I hope you've all had a great week learning all about CESM. I'm not going to teach you anything new about CESM itself, but rather show you how you can continue learning to use it on your very own laptop via a project we call "CESM-Lab". Feel free to follow along as we go, pausing this video as needed to do the required steps. This will require downloading some software called Docker onto your laptop or desktop, and downloading a few gigabytes of data to get started, so it could take a while depending on your internet speeds. Finally, everything I'm going to cover is pretty new technology for us – and it's still in development, so we'd *love* feedback, so we can improve this moving forward for the community.



Overview

- What *is* CESM-Lab?
- Downloading Docker
- Downloading CESM-Lab
- Launching CESM-Lab:
 - GUI
 - Command-line
 - Connecting via a Browser
- Tutorial Walk-Through
- Using the Terminal
- Limitations
- Future Plans

Here's a quick overview of what we'll cover today – starting with a brief look at what CESM-Lab actually *is*. After that, I'll discuss how to download Docker, and how to get the model software itself. Then I'll give brief examples of how to launch it, on both Windows and Mac / Linux systems. Then we'll spend a few minutes looking at a *very* basic Jupyter-based tutorial we have included for now, and talk a little bit about how we expect to expand on this in the future. And, since some of you might be 'old school' like myself, I'll discuss how to use the terminal included, vs the Jupyter interface. And finally, we'll close out with a quick discussion of some limitations, since obviously the current target here is personal systems, which have considerably less power than supercomputers like Cheyenne! Last but not least, I'll briefly cover some future plans.

What is CESM-Lab?

CESM-Lab: *Containerized* environment with CESM + Jupyter Lab


+


What is a <i>container</i> ?	
Portable	Run <i>almost</i> anywhere (Mac, Windows, Linux)
Preconfigured	No installation or porting required!
Standardized	All users get the <i>same environment</i>

A simple, easy-to-use platform for CESM teaching & research

NCAR
UCAR
CESM Tutorial 2021 – Intro to CESM-Lab

So, what *is* CESM-Lab? Simply enough, it's just a preconfigured environment with both CESM and Jupyter Lab. It's portable, and ready to run pretty much anywhere. And containers are the key technology here – they're a technology that lets you package up a whole environment for an application like CESM, and provide it in this simple, portable form. For us, this isn't just the model source code, but also all the libraries, compilers, tools and more that go into having a usable environment for modeling. Given the rapidly growing interest in Jupyter notebooks, and the growth of things like the Pangeo project and NCAR's Earth System Data Science (ESDS) initiative, we can also include a full stack of Python packages needed for visualization and analysis.

Best of all, this means anything we provide in this environment that works for one person, will work for another! No struggling to add new libraries or dependencies – it just works. You can focus on shareable science, instead of wrestling with conda environments, or installing tools. This also allows for the development of *common materials* for training, like we'll see and use later on. If you're *not* familiar with Jupyter, don't worry – you can use a traditional 'terminal' environment in multiple ways, too, and I'll cover that a little later on.

(Earth image from: Andrew Colvin -
[https://commons.wikimedia.org/wiki/File:Earth_\(blank\).png](https://commons.wikimedia.org/wiki/File:Earth_(blank).png))

Downloading Docker

<https://www.docker.com/products/docker-desktop>

Docker Desktop

The fastest way to containerize applications on your desktop

Mac with Intel ChipMac with Apple Chip

MOST COMMON

Also available for [Windows](#) and [Linux](#)

By downloading this, you agree to the terms of the [Docker Software End User License Agreement](#) and the [Docker Data Processing Agreement \(DPA\)](#).

(Free, but you'll need 'admin' privileges on your system to install!)

NCAR
UCAR | CESM Tutorial 2021 – Intro to CESM-Lab

Docker is a common 'container' application – it's free to use, easy to install (you'll need admin privileges though!), and user-friendly. If you go to that link, it'll show you what it thinks is right for your system – in my case, I'm on a Mac, so it shows me these options. If you're following this presentation at your own pace, I'd recommend pausing this presentation now, installing the Docker software, and then resuming.

Downloading CESM-Lab

Open a terminal or command-prompt, and use the command:

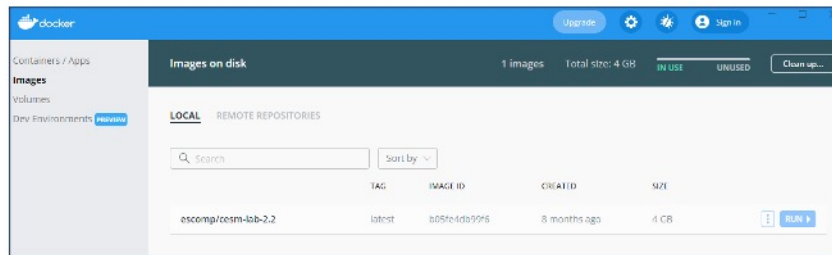
```
docker pull escomp/cesm-lab-2.2
```

This image is approximately 4GB in size – this includes a Linux OS, compilers, libraries, CESM code, Jupyter Lab, Pango environment, text editors and more!

OK, great – if you have Docker, now we can download the CESM-Lab image. Open a terminal or command prompt and run this ‘docker pull’ command shown above - this will download all the CESM software, This image is about 4 gigabytes in size, so once again, if you’re following along to this video at your own pace, I recommend pausing it until it’s complete.

Running CESM-Lab – GUI

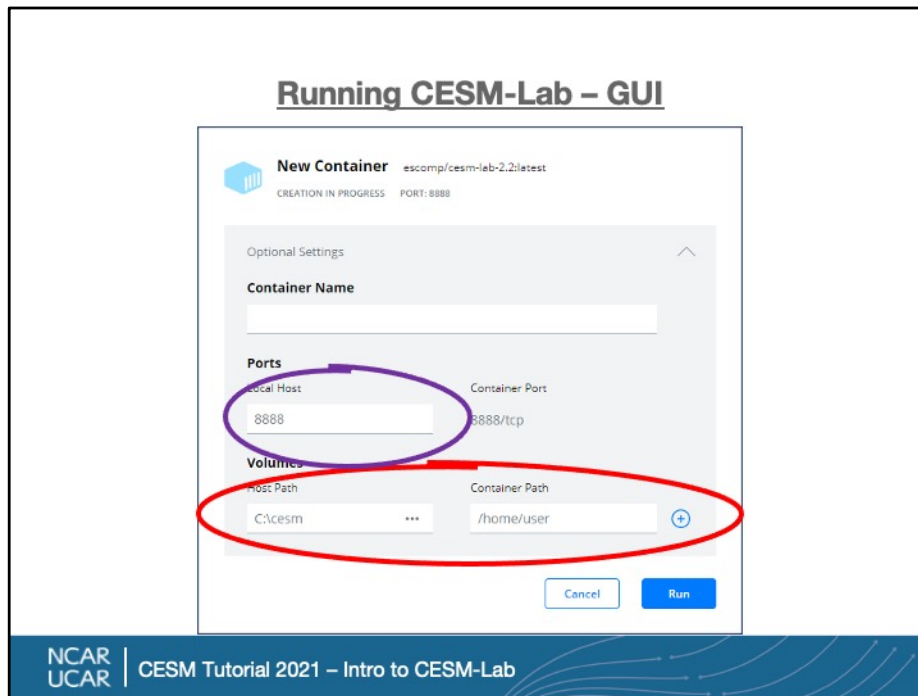
On Windows or Mac, launch the Docker dashboard...



You should see the downloaded 'cesm-lab-2.2' image.
Now just click the 'run' button.

Now let's talk about running the container. Both Mac and Windows offer a GUI for Docker – you can use that, *or* the command line. Ultimately, both work the same. I've seen more people using the GUI on Windows, so these screenshots are from Windows, and the command-line ones will be from a Mac system, but it's just the same two options you need to set.

So once you launch the GUI 'dashboard' (often at the top of your Mac screen, or in the Windows start menu), you should see the container listed among the images we have. From here, we're just going to go click on the 'run' button.



Now, to be fully functional, we need to provide two settings to Docker – the first is a ‘port’, circled here in purple. This just lets us communicate with the containerized Jupyter environment via a browser. Generally we just use 8888, and only need to change it if another piece of software is already listening to that port.

The second option we’re providing is a ‘mount point’, circled here in red. What this does is make a directory of our choosing on our system visible to the containerized environment, letting us ensure our all work –model cases, output data, scripts, etc!- are all saved when you exit the container. (By default, containers don’t update their image – this is great for making sure you have a standard, working copy, but it does mean we need to map in a location.)

Once we set these values, click run, and you’re off and running!

Running CESM-Lab: Command-Line

`docker run -v ~/cesmwork:/home/user -p 8888:8888 escomp/cesm-lab-2.2`


Output:

```

bdobbins@cgdm-demo ~ % docker run -v ~/my_cesm_directory:/home/user -p 8888:8888 escomp/cesm-lab-2.2
[I 03:21:20.983 LabApp] [nb_conda_kernels] enabled, 1 kernels found
[I 03:21:21.016 LabApp] Writing notebook server cookie secret to /home/user/.local/share/jupyter/runtime/notebook_cookie_secret
[W 03:21:22.672 LabApp] All authentication is disabled. Anyone who can connect to this server will be able to run code.
[I 03:21:25.889 LabApp] JupyterLab extension loaded from /srv/conda/envs/default/lib/python3.7/site-packages/jupyterlab
[I 03:21:25.890 LabApp] JupyterLab application directory is /srv/conda/envs/default/share/jupyter/lab
[I 03:21:25.906 LabApp] Serving notebooks from local directory: /home/user
[I 03:21:25.906 LabApp] Jupyter Notebook 6.1.4 is running at:
[I 03:21:25.906 LabApp] http://f4ad71b6df47:8888/
[I 03:21:25.907 LabApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 03:21:25.917 LabApp] No web browser found: could not locate runnable browser.

```

(The output says 'no web browser' found – but don't worry about that!)


 | CESM Tutorial 2021 – Intro to CESM-Lab

If you've followed the previous two slides, this is more of the same, only via the command-line – this approach works on Mac, Windows and Linux. If you're using CESM, you're *probably* using a terminal at some point, so I highly recommend trying things this way, even if you're more at ease with the GUI mode. But if you've already launched it using the GUI, skip this step for now, and just try it next time!

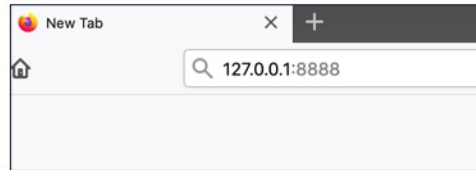
Just the same as the GUI method, we need to tell Docker two important things – a directory to 'map' into the container, and a 'port' to connect to the Jupyter server we're running. For the directory, specified with the '-v' option, as seen in red here, we can specify whatever we want – if it doesn't exist, it'll be created. In the example here, I'm choosing to use a directory called 'cesmwork' in my home directory. This gets mapped to '/home/user' in the container. For ports, using the '-p' option, seen in purple here, I'm using the defaults – mapping port 8888 from my system to port 8888 of the container, where the Jupyter server listens. Again, if you are running Jupyter already on your system, that port *might* be in use, and in that case you can switch the first number to something different – eg, 9999:8888.

Press enter, and you'll see some output like the above – one thing that admittedly causes some confusion is the 'Jupyter log' messages that say a browser can't be found. We can ignore those; we're running Jupyter inside the container, and using our own browser to connect. We'll change this in a future version to be more clear!

Running CESM-Lab: Connecting via a Browser

Now we have it running, so we just need to *connect*.

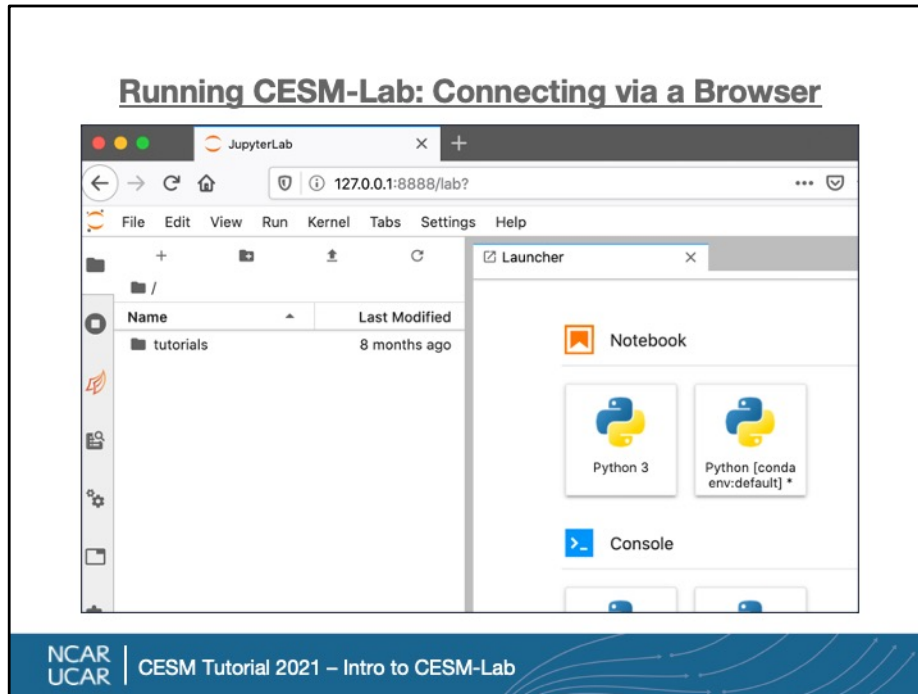
Open a browser and point to our address & port:



(Our local system is always "127.0.0.1", but if you used a different port to launch the container, use the same first number here.)

At this point, if you've done either of the prior steps, you should have a running container ... but not really much to 'show' for it yet. That's just because it runs as a background process – to *do anything*, we need to connect to it. We'll do that by opening up a web browser, with Firefox shown here, and connect to our local system, on the port we provided in those options we used when launching it.

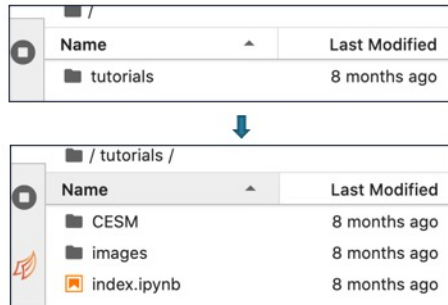
So, open up the browser, go to 127.0.0.1:8888 (or whatever port you used instead of 8888), and press enter... and you should be greeted by...



The Jupyter Lab welcome screen! Of course, there's not a lot to see here yet, especially if we didn't map in an existing directory, but that's OK. We *do* have this 'tutorials' directory, which is what we're going to explore next.

• Tutorial Walk-Through

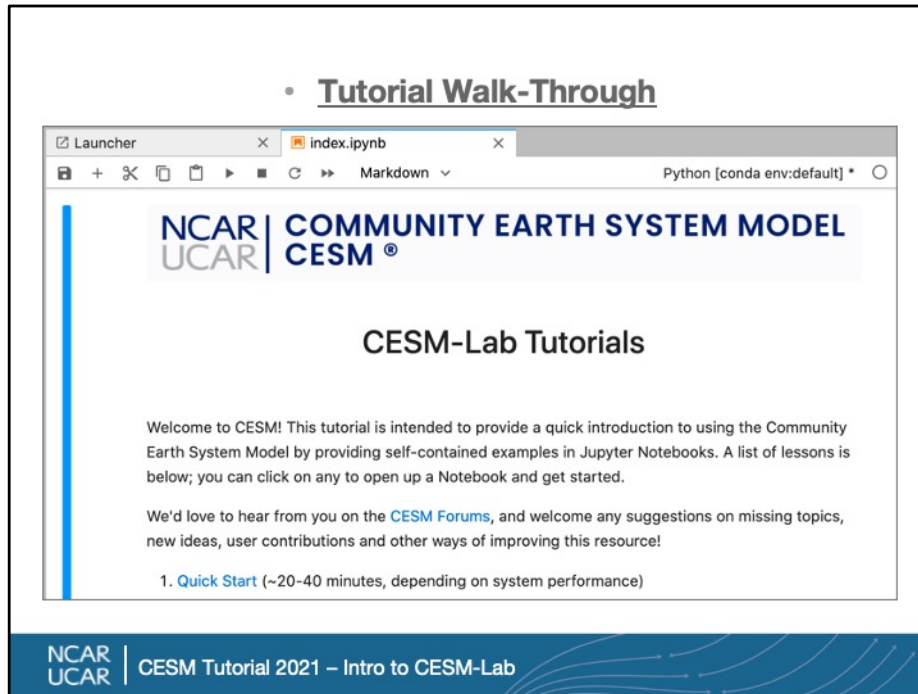
OK, now that we're in Jupyter, let's access the tutorial – on the left side of the browser window, selection the 'tutorials' directory:



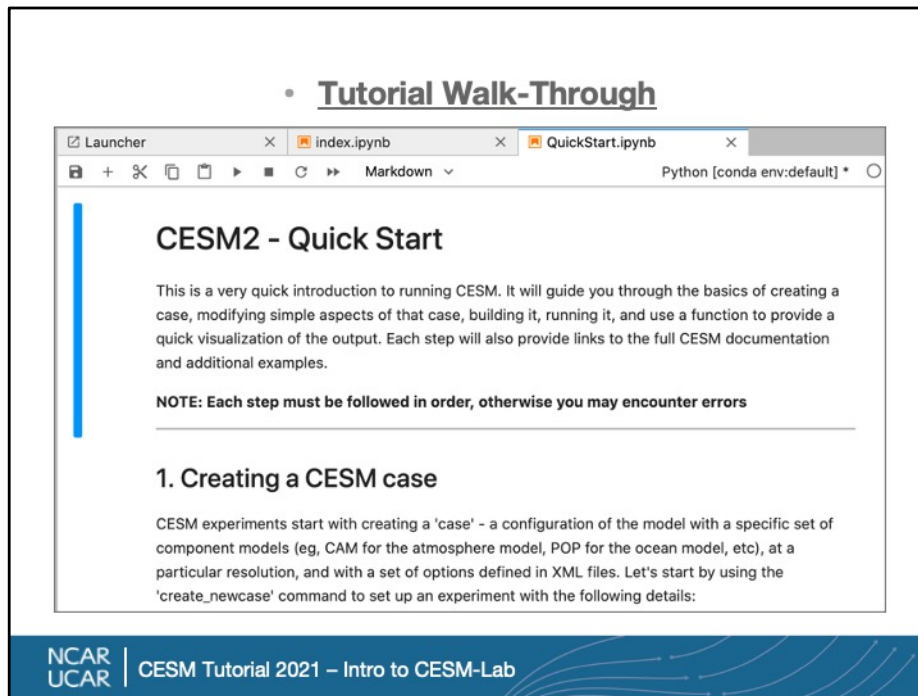
And then click on the index.ipynb file...

Now, let's pull up our tutorials – to do this, on the left side of the Jupyter screen, double click on the tutorials. That should open the folder, and show you the second image above, with the file 'index.ipynb' – this is our main page for the tutorials. Double click on it, and open up the main page.

• Tutorial Walk-Through



This is the main page for what will be a set of tutorials – though, at the moment, there’s really just a quick start. We will be improving this in the future, but that quick start is still going to give you a basic overview of using Jupyter with CESM. There’s also a link to the forums for asking questions, but for now, let’s click on the quick start link shown at the bottom of the screenshot above.



For anyone unfamiliar with a Jupyter Notebook, this is an example of what one looks like – it has cells (seen by the blue line on the left) of text or documentation, and, when you scroll down, you’ll see executable cells as well. You COULD run this whole notebook, and it would proceed through each runnable cell, creating a case, setting it up, building it, running it, running it some more, and providing a quick visualization of the result... but it’s better to go step by step. We’ll look at the first cell in the next slide.

• Tutorial Walk-Through

The grey section below is the first runnable cell.

The command to create this configuration is given below, using 'quickstart_case' as the casename. Note that given the size and nature of this run, it's not a scientifically supported configuration, but it's still useful for learning how to run the model.

Run the cell to issue the command:

```
[ ]: create_newcase --case ~/quickstart_case --compset QPC4 --res f45_f45_mg37 --run-unsup
```

You can run it via selecting it, then pressing shift+enter.
It functions the same as it did on Cheyenne!

Here's our first runnable cell – it's a pretty standard 'create_newcase' command, with just a few small differences. For one, we don't need to navigate to a particular directory here; we know we're using CESM 2.2, since that's the image we downloaded. And two, you'll note we're using a really coarse resolution, and a 'QPC4' compset, which is an aquaplanet with CAM4 physics. That's a pretty low resolution, old-physics setup, so why are we using it? Mostly because it's fast – you've been running on Cheyenne, on multiple nodes, whereas now we're running on a laptop, most likely. This is something we can reasonably run with limited memory and time. Note that the container is NOT limited to these low-res modes – if you have a workstation or server with more than 128GB of RAM and a lot of processors, you could use this same container to run a fully-coupled B compset at 1-degree resolution, too. But this is good for experimenting, and a lot of useful models, like single-column cases or simple models, run fairly well on laptops.

(Finally, one note for people who *are* familiar with Jupyter is that we're actually doing something non-standard here, for the purposes of this very introductory lesson. That is, normally to run code in a Jupyter cell, you'd want to specify it's to be run in a bash prompt, or we run Python code directly – here, we're

cheating a bit, so that we're running the *exact same command* we'd use in a terminal inside the Jupyter environment. This only works for single commands, and later tutorials will cover more correct ways to do things.)

• Tutorial Walk-Through

```
[1]: create_newcase --case ~/quickstart_case --compset QPC4 --res f45_f45_g37 --run-unsupported
Compset longname is 2000_CAM40_SLND_SICE_DOCNAQP3_SROF_SGLC_SWAV
Compset specification file is /opt/ncar/cesm2/components/can/cime_config/config_compsets.xml
Automatically adding SIAC to compset
Automatically adding SESP to compset
Compset forcing is 1972-2004
ATM component is CAM can4 physics:
LND component is Stub land component
ICE component is Stub ice component
OCN component is DOCN analytic_aquaplanet_sst - option 3
ROF component is Stub river component
GLC component is Stub glacier (land ice) component
WAV component is Stub wave component
IAC component is Stub iac component
ESP component is Stub external system processing (ESP) component
Pes specification file is /opt/ncar/cesm2/components/can/cime_config/config_pes.xml
Compset specific settings: name is SSTICE_DATA_FILENAME and value is $DIR_LOC_ROOT/ata/cam/sst/sst_had0181_bc_1x1_200
0c1ino_c180511.nc
Could not find machine match for 'ec8fe5451a7d' or 'ec8fe5451a7d'
Machine is container
Pes setting: machine match is container
Pes setting: grid is @4x5_lnull_o14x5_rnull_gnull_wnull_znull_m9x3v7
Pes setting: compset is 2000_CAM40_SLND_SICE_DOCNAQP3_SROF_SGLC_SWAV_SIAC_SESP
Pes setting: tasks is {'NTASKS_ATH': -1, 'NTASKS_LND': -1, 'NTASKS_ROF': -1, 'NTASKS_ICE': -1, 'NTASKS_OCN': -1,
'NTASKS_GLC': -1, 'NTASKS_WAV': -1, 'NTASKS_CPL': -1}
Pes setting: threads is {'NTHRDS_ATH': 1, 'NTHRDS_LND': 1, 'NTHRDS_ROF': 1, 'NTHRDS_ICE': 1, 'NTHRDS_OCN': 1, 'NTHRDS_GLC': 1,
'NTHRDS_WAV': 1, 'NTHRDS_CPL': 1}
Pes setting: rootpe is {'ROOTPE_ATH': 0, 'ROOTPE_LND': 0, 'ROOTPE_ROF': 0, 'ROOTPE_ICE': 0, 'ROOTPE_OCN': 0, 'ROOTPE_GLC': 0,
'ROOTPE_WAV': 0, 'ROOTPE_CPL': 0}
Pes setting: pstrid is {}
Pes other settings: {}
Compset is: 2000_CAM40_SLND_SICE_DOCNAQP3_SROF_SGLC_SWAV_SIAC_SESP
Grid is: @4x5_lnull_o14x5_rnull_gnull_wnull_znull_m9x3v7
Components in compset are: ['cam', 'slnd', 'sice', 'docn', 'srof', 'sglc', 'swav', 'siac', 'sesp']
No project info available
No charge_account info available, using value from PROJECT
cesm model version found: cesm2.2.0
Batch_system_type is none
Creating case directory /home/user/quickstart_case
This component includes user_mods /opt/ncar/cesm2/components/can/cime_config/usermods_dirs/aquap
Adding user_mods directory /opt/ncar/cesm2/components/can/cime_config/usermods_dirs/aquap
```

NCAR
UCAR

CESM Tutorial 2021 – Intro to CESM-Lab

After running the cell, you should see all the usual CESM output – this is identical to what you see when running via a command line, and the only real difference is that in the container, we detect how many processors you have, and use that for the PE count. On modern laptops, this typically ranges from 2-8 cores, but on Cheyenne this would use 36 (a single node).

• Tutorial Walk-Through

2. Modifying a CESM case

Now let's make a simple modification to the case via two easy steps:

1. Change to our new case directory
2. In that directory, change the duration of the run (in model-time)

These changes can be made via modifying the XML files that control CESM's behavior. We could do this by hand, but the preferred way of doing it is to use the 'xmlchange' command in the case directory. By default, newly configured cases are set up to run for 5 model days - since we're just experimenting, we'll change it to 3 model days instead, for even faster runs. Run the following cells to execute the two commands:

```
[2]: cd ~/quickstart_case  
/home/user/quickstart_case
```

```
[3]: xmlchange STOP_OPTION=ndays,STOP_N=3
```

Since the default STOP_OPTION is already 'ndays', it wasn't strictly necessary to include that in the command, but by being explicit we avoid any confusion later if we had changed it previously to 'hours' or 'years'.

We don't see any output by 'xmlchange', but we can use another tool, 'xmlquery', to double-check that we have the new values:

```
[4]: xmlquery STOP_OPTION,STOP_N
```

```
Results in group run_begin_stop_restart  
STOP_OPTION: ndays  
STOP_N: 3
```

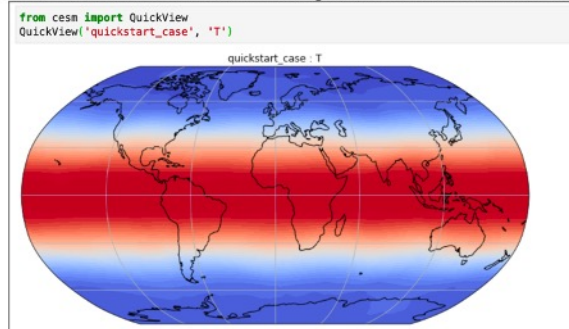
NCAR | UCAR | CESM Tutorial 2021 – Intro to CESM-Lab

As you progress through this quick start, you'll do a few basic modifications, again using commands you're already familiar with. In this case, we're setting up our run for only 3 model days. On my laptop, a 2018 Mac with 4 cores, it takes about 20 seconds per model day to run, so this should take about a minute.... But before we run, we still need to build the model. That alone can take about five minutes.

So if you're working along with this at your own pace, I recommend pausing this here, and following the rest of the steps in the Quick Start. All told it should take between 20-40 minutes to do everything, which includes running for a full month, then plotting some output. Obviously the actual amount of time will depend on your system's processors, and how long it takes to download a little bit of data.

- Tutorial Walk-Through

If you run through *all* the steps, you'll eventually end up with a quick plot like below. This is an *example* of how to mix documentation, code and analysis/visualization in Jupyter.

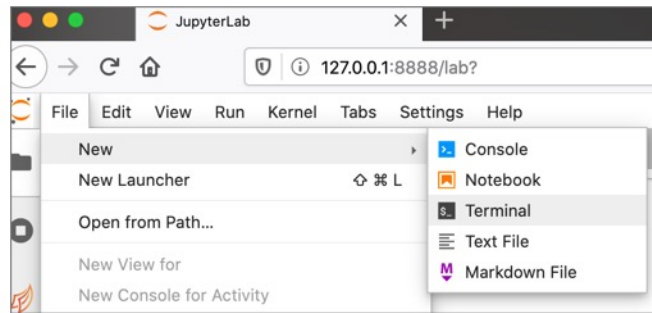


Also look at NCAR's ESDS Initiative: <https://ncar.github.io/esds/>

OK, great! I hope you made it through all the steps in the quick start by now – if you have, you should see two plots in your Notebook, the first looking just like the one above. At this point, you've run a full CESM case on your own system, via a Jupyter notebook, and did a quick Python-based visualization of the output! This is, as the slide says, just an *example* of how you can combine documentation, code and analysis or visualization in Jupyter notebooks. They're a really great way of organizing and displaying work, and you can share CESM-Lab notebooks with others, and they can replicate your work. This is just the very tip of the iceberg on CESM analysis and visualization, though - making more complex visualizations is an ever-evolving topic, and NCAR recently launched an Earth System Data Science (ESDS) initiative that I'd encourage you to look into if this interests you, with the link above. It's a growing community offering expertise and ideas on how to use Python and notebooks to do data science.

Using the Terminal

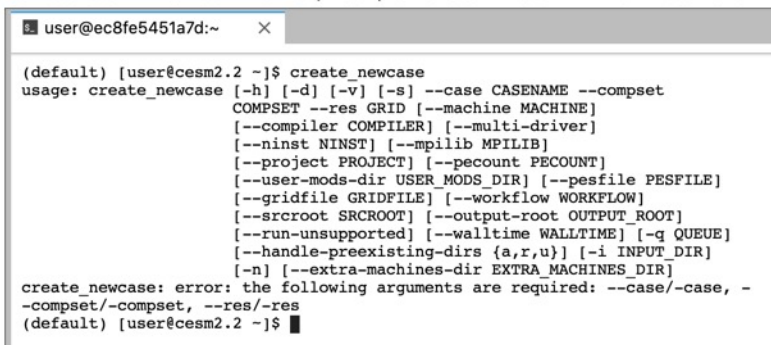
Within Jupyter, you can *also* launch terminals:



If you've gone through the whole Quick Start notebook, and find that you prefer using a terminal over a Notebook, no problem! There are actually a few ways of launching a terminal in the container – for now, we'll just cover the Jupyter method. To do this, just use the File -> New -> Terminal option.

Using the Terminal

This looks and functions just like any other Linux terminal – you can do commands like 'ls', 'cd', and edit files with text editors.



```
user@ec8fe5451a7d:~  
(default) [user@cesm2.2 ~]$ create_newcase  
usage: create_newcase [-h] [-d] [-v] [-s] --case CASENAME --compset  
COMPSET --res GRID [--machine MACHINE]  
[--compiler COMPILER] [--multi-driver]  
[--ninst NINST] [--mpilib MPILIB]  
[--project PROJECT] [--pecount PECOUNT]  
[--user-mods-dir USER_MODS_DIR] [--pesfile PESFILE]  
[--gridfile GRIDFILE] [--workflow WORKFLOW]  
[--srcroot SRCROOT] [--output-root OUTPUT_ROOT]  
[--run-unsupported] [--walltime WALLTIME] [-q QUEUE]  
[--handle-preexisting-dirs {a,r,u}] [-i INPUT_DIR]  
[-n] [--extra-machines-dir EXTRA_MACHINES_DIR]  
create_newcase: error: the following arguments are required: --case/-case, -  
-compset/-compset, --res/-res  
(default) [user@cesm2.2 ~]$
```

This browser-based terminal is a full environment, using the bash shell by default, and will let you navigate your files, run commands, edit files and use standard Linux utilities just as you would on any other system like Cheyenne. As you can see in the screenshot above, we can also directly call CESM commands, like 'create_newcase', without having to navigate to a particular CIME directory, too.

Limitations

Memory: Some cases require more than your system may have!

Example #1: QPC6 @ f09_g17 needs ~15GB of RAM

Example #2: B1850C4_Tutorial @ f45_g37 needs ~8GB of RAM

Data: Some cases, particularly some land ones, need *lots* of data!

Example #1 : F2000climo @ f09_g17 needs ~6GB of data

Example #2: I2000CIm50Sp @ (any res) needs ~400GB of data!

Obviously, switching from running on a large cluster to a laptop, desktop or server means you have fewer resources – if your laptop only has 8GB of RAM, you won't be able to run the CAM6 aquaplanet config at 1-degree, since it needs about 15GB of RAM. But if you install it onto a workstation or server with 32GB, that should work fine. (Our QPC4 @ f45_f45_mg37 only needed 0.6GB!). Another interesting data point is this B1850C4_Tutorial compset, which is a coupled run but using less-intensive CAM4 physics, and that only needs about 8GB of RAM to run, letting you run a coupled model *on some newer laptops*.

Switching to data, CESM downloads input data that it needs it for a case, typically when submitting it for the first time – for many cases, it's relatively small amounts, but some land cases require hundreds of gigabytes of data! Granted, it only needs to download this once, saving it in the directory you mounted in, and then it can be used for any number of runs of that compset, but be aware that you may need lots of disk space.

Finally, not listed here, it should be obvious that performance also drops greatly if you go from running on hundreds of processors to just a few. The key here, though, is that these are all *hardware* limitations of whatever system you're

using; the environment is a fully functional version of CESM. And we'll talk now, on the next slide, a bit about how we're using the same technology that works on your laptop to help you run this on university clusters in the near future.

Future Plans

More hardware options:

- Run on University clusters via an HPC container
- Run in the cloud (pay as you go)
- 'Offload' from laptop to NCAR or Cloud

More training material:

- We'll be adding more tutorials in the near future!

So, if you think, “Hey, this is pretty neat, but I want to run large, fully-coupled runs on my university cluster, but have trouble porting CESM..”, then drop me a line. We’re working on another container specifically for this, and we’d love to coordinate with you on trying it out!

We also have the ability to run on AWS, and soon, Azure, and hope to make those public soon – this is great if you *don't* have your own clusters, but it gets expensive fast compared to local systems.

Another thing we’re looking at doing in the future is being able to ‘offload’ a run from this laptop/desktop CESM-Lab, to NCAR or the Cloud if you have accounts in either. This lets you do everything, even build cases, via a local system... but then run on bigger ones with lots more resources.

Finally, we’re hoping to expand the tutorials in the near future – ideally offering a full set of training examples for people to learn on, including running the model, analyzing output and visualizing results.

Thank You!

Hope you've enjoyed learning about running CESM-Lab, and we hope it helps you get up and running fast with CESM. If you have comments, questions or issues, we're happy to talk with you!

Email:
bdobbins@ucar.edu

Thanks again for your time, and I hope this helps you get started with CESM, and Jupyter. Let us know any feedback you have so we can improve this to make it even better suited for your needs!