TACC – Apptainer + PyLaucher Instructions

Below is what I did to get PyLaucher and Apptainer to work on TACC. It probably can be simplified, but this is what worked for me. I created my image using Docker container, but TACC uses the module Apptainer for container implementation instead of Docker because “the Docker run time is not secure on systems where users are not allowed to have ‘escalated privileges’”. Also, for my example, I use PyLaucher to specify a single core per job (or, in our case, simulation condition). I did not need to run multi-threaded, MPI, or GPU-accelerated tasks.

Note: Also, the guy who made PyLauncher, Victor Eijkhout, seems to be currently working on the module, so these instructions work as of March 23, 2025, but maybe it will not work with future updates to PyLauncher. PyLauncher does seem to be a work in progress.

1. **Set up your local project directory** 
   1. You can do this differently, but for my project, I created a folder for my TACC-related files and then a subfolder for my Docker image-related files. Below is a breakdown of the files I used for my test; I will go into more detail about some of these files in the steps that follow. [here are my files related to this test](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/tree/main/TACC_files/test).
      1. The “[test](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/tree/main/TACC_files/test)” folder contains:
         1. [powstudcatmod\_test](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/tree/main/TACC_files/test/powstudcatmod_test)
            1. a folder that contains the files that are needed to build the Docker image

* + - 1. [files\_pylauncher\_test.R](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/files_pylauncher_test.R" \o "files_pylauncher_test.R)
         1. A R file that makes files necessary for PyLauncher to work, “command\_lines\_test” and “powstudcatmod\_test/ [pyl\_id\_values\_test.csv](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/powstudcatmod_test/pyl_id_values_test.csv)”.

* + - 1. [command\_lines\_test](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/command_lines_test" \o "command_lines_test)
         1. a file that PyLauncher needs, it is essentially a queue of jobs to send to PyLauncher. Each line is a job that will run on one core. All the commands on each line are related to one job. You will notice that the only thing that changes across each job is the second argument of –args. Example: --args PYL\_ID # "/home/r-environment/output", where # is the condition I want to run. You can modify these commands in [files\_pylauncher\_test.R](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/files_pylauncher_test.R) if you need to. Below is a breakdown of what is happening in each line:

“apptainer run”: tells Apptainer to run a container.

“--bind /scratch/08147/bethanyh/pylauncher\_out:/home/r-environment/output”: The --bind option maps a directory from your host system (TACC) to a directory inside the container (These are also called bind mounts). “/scratch/08147/bethanyh/pylauncher\_out” is the directory on TACC where I want to save my output and “/home/r-environment/output” is a folder inside the container where the R script will save the results.

“/scratch/08147/bethanyh/powstudcatmod\_test\_v3.sif”: is the path to the container image file (.sif) on TACC. After the container is pulled from Docker (“apptainer pull docker://bethanyhamilton/powstudcatmod\_test:v3” in [jobfile.sh](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/jobfile.sh)), TACC saves the image as a .sif file in whatever directory you are in when you sbatch jobfile.sh.

“R -e "source('/home/r-environment/run\_sim\_study.R')"”: Launches R inside the container and evaluates the command in quotes.

“--args PYL\_ID 1 "/home/r-environment/output"”: command-line arguments that get passed to [run\_sim\_study.R](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/powstudcatmod_test/run_sim_study.R). The Rscript takes three arguments: 1) PYL\_ID 2) which condition to grab 3) the output file path on TACC. Only the second argument will change across jobs, as can be seen.

* + - 1. [jobfile.sh](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/jobfile.sh)
         1. My slurm batch script that I run on TACC. Please note:

#SBATCH --ntasks-per-node 48

This tells PyLauncher how many cores are in each node.

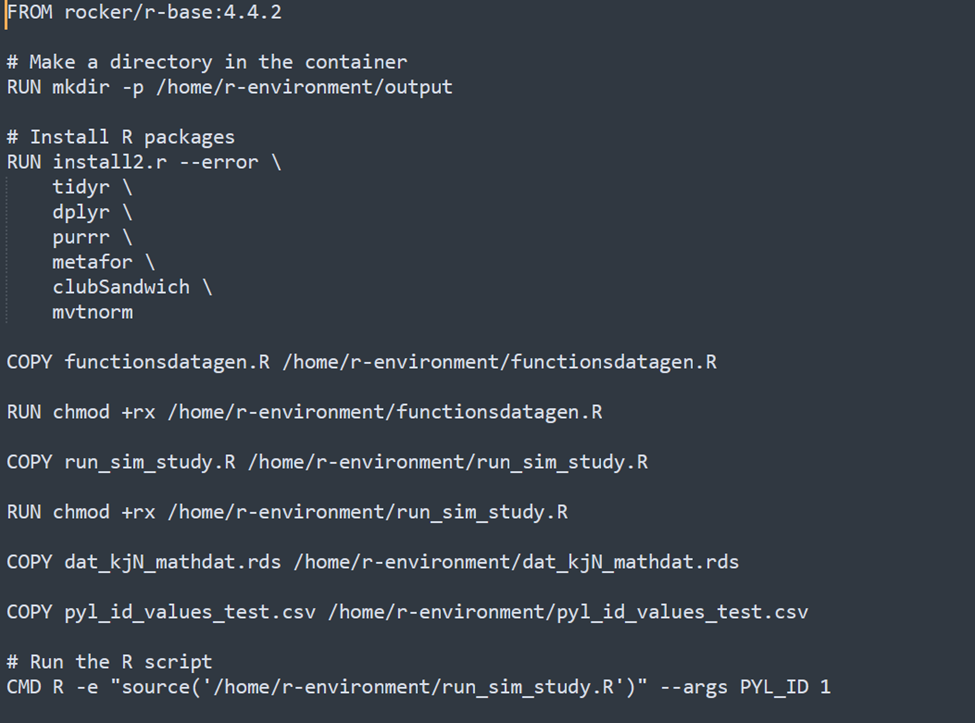
Make sure to load the same modules I did; they are all necessary.

“apptainer pull docker://bethanyhamilton/powstudcatmod\_test:v3” this command pulls my image off of Docker Hub. Apptainer on TACC saves the image as an .sif file in whatever directory you are in when you run “sbatch path\_to\_jobfile/jobfile.sh”.

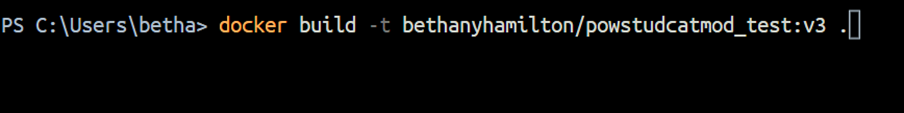
“python3 /work2/08147/bethanyh/stampede3/test123/pmcmlauncher.py”: runs the PyLauncher script.

* + - 1. [pmcmlauncher.py](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/pmcmlauncher.py)
         1. A python script that imports PyLauncher and implements a function from that module called ClassicLauncher(). The first path is the location of the [command\_lines\_test](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/command_lines_test) file on TACC. The “workdir” argument is where your output files will go on TACC. I believe the debug argument is not necessary, but helps with troubleshooting errors later in the output files of PyLauncher.
    1. The “[powstudcatmod\_test](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/tree/main/TACC_files/test/powstudcatmod_test)” folder contains:
       1. [Dockerfile](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/powstudcatmod_test/Dockerfile)
          1. A file used to build the Docker image.
       2. [run\_sim\_study.R](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/powstudcatmod_test/run_sim_study.R)
          1. My Rscript that runs the simulation. This file reads in the arguments from [command\_lines\_test](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/command_lines_test) as described earlier. Then, it sources the other files necessary for my simulation which are stored in “/home/r-environment/”, the path where these files will be found on the Docker image. I do not need to run things in parallel since, in this set up, this Rscript will run just one condition per core.
       3. [pyl\_id\_values\_test.csv](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/powstudcatmod_test/pyl_id_values_test.csv)
          1. A file that results from [files\_pylauncher\_test.R](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/files_pylauncher_test.R) script. It contains different combinations of the simulation parameters I want to run. Each row is a condition of my simulation. The last column, PYL\_ID, is an identifier that [run\_sim\_study.R](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/powstudcatmod_test/run_sim_study.R) uses to extract the correct condition I want to run. The file [command\_lines\_test](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/command_lines_test) passes the PYL\_ID argument to [run\_sim\_study.R](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/powstudcatmod_test/run_sim_study.R) and then [run\_sim\_study.R](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/powstudcatmod_test/run_sim_study.R) uses the [pyl\_id\_values\_test.csv](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/powstudcatmod_test/pyl_id_values_test.csv) to subset the corresponding row to the grab the simulation parameters I want to run for that condition.
       4. [dat\_kjN\_mathdat.rds](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/powstudcatmod_test/dat_kjN_mathdat.rds) and [functionsdatagen.R](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/powstudcatmod_test/functionsdatagen.R)
          1. Files that are only relevant to my simulation, so don’t worry about them. You just need them if you want to recreate this example.

1. **Set up your environment on TACC**
   1. After you SSH into TACC (using whatever SSH client you prefer), navigate to $WORK (cd $WORK) and create a project folder in $WORK. For my test, I called mine “test123” (mkdir test123).
   2. Then, navigate to $SCRATCH (cd $SCRATCH), and create a folder for your output files. For my test, I called mine “pylauncher\_out” (mkdir pylauncher\_out).
2. **Get Docker Desktop set up**
   1. Download Docker Desktop for your OS.
   2. Login to Docker from the terminal in Docker Desktop:
      1. If you don't have a Docker Hub account, create one at Docker Hub.
      2. Open the terminal on Docker Desktop and run, “docker login”. This will prompt you for your username and password to authenticate with Docker Hub.
3. **Prepare simulation files that will go on your Docker image**
   1. In a local project folder, put all the files you need to run your simulation.
   2. Set up your run\_sim.R file as I did, specifying the path where the simulation files will be located on the container: “/home/r-environment/” in my case.
   3. Prepare **“**[pyl\_id\_values\_test.csv](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/powstudcatmod_test/pyl_id_values_test.csv)” using “[files\_pylauncher\_test.R](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/files_pylauncher_test.R)”
      1. I just wanted to do a quick test, so I ran only 1 out of 1 batch of 12 conditions with 3 replications each. When modifying this, be sure to create PYL\_ID as I did, though.
4. **Build your docker container locally**
   1. **Create the Dockerfile**:
      1. In your project directory, create a Dockerfile. This file does not need a file extension. Example from my test run:



* + - 1. Above, I installed the R version that I am using for my sim.
      2. Then, I make the directories I need to run the sim and mount to TACC-- “r-environment” is where all my sim-related files go, and “output” is where my data will spit out (I needed to create a separate directory for my data because for some reason the files in “r-environment” get erased when I mount the image and spit out the data in the same directory).
      3. Then, I installed the packages I needed for the sim.
      4. “Copy” is used to copy the files I need for the simulation that are stored locally to the docker image.
      5. chmod +rx <filename> is “change mode” and it modifies the file and directory permissions. And “+rx” means add read and execute permissions.
  1. **Build the Docker image**:
     1. In your terminal, navigate to the folder containing the Dockerfile and all your sim-related files using “cd <path to docker-related files>”.
     2. Then use “docker build -t account/image\_name:version .” to build the image. For example:



* + - 1. This will build the image and tag it with the version (a necessary step if you want to push to Docker hub). In my case, the image is called powstudcatmod\_test, and the version is v3. If you haven't already tagged the image with your Docker Hub username, run: docker tag docker-image-name:your-tag your-dockerhub-username/docker-image-name:your-tag to tag it.
  1. **Push the image to Docker Hub**:
     1. To push the image to Docker Hub, run: docker push your-dockerhub-username/docker-image-name:your-tag. This command uploads your image to Docker Hub, which can be pulled from any machine.

1. **Upload necessary files to TACC**
   1. Using whatever data transfer method you prefer (TACC has [these](https://docs.tacc.utexas.edu/basics/datatransfer/) suggestions, I used WinSCP as "~\Pusto lab\CHTC\condor steps.docx" suggested, and I like it), navigate to the “test123” folder in the $WORK directory (see step 2). There, you will place the [command\_lines\_test](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/command_lines_test), [jobfile.sh](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/jobfile.sh), and [pmcmlauncher.py](https://github.com/bethanyhamilton/PowerMeta_StudyCatMod/blob/main/TACC_files/test/pmcmlauncher.py) files.
2. **Submit the job**
   1. SSH into TACC again.
   2. Go to the $SCRATCH directory (cd $SCRATCH).
   3. Submit slurm job file using sbatch: (sbatch /work2/08147/bethanyh/stampede3/test123/jobfile.sh)
   4. Sit back and relax or obsessively check the status using: squeue -u command (squeue -u bethanyh).
3. **Extract Files**
   1. If you have successfully executed the above steps, then your data and some “Success” files should pop up in the pylauncher\_out folder in $SCRATCH. If it didn’t, then you may only see some “expire” and “out” files.
   2. NOTE: A quirky feature of PyLauncher is that you need to clear all of our output files in pylauncher\_out/ and the parent directory (/scratch/08147/bethanyh in my case) between runs; otherwise, it will result in an error. In particular, you must remove the docker image (file with extension .sif) from the parent directory of /pylauncher\_out/.

References:

* <https://containers-at-tacc.readthedocs.io/en/latest/index.html>
* <https://docs.tacc.utexas.edu/software/pylauncher/>
* <https://docs.tacc.utexas.edu/hpc/stampede3/>
* <https://github.com/TACC/pylauncher>
* "~\Pusto lab\CHTC\condor steps.docx"

Potential errors/Troubleshooting Notes:

-Make sure all your files have Unix encoding. For example, line breaks are different in Windows and Unix. There are many ways to do this, but I use Sublime text for a text editor. Here you can create a new file. Then, before typing, go to View > Line Endings > Unix, and begin adding syntax to your file. Another example that came up for me was in the PyLauncher documentation (https://docs.tacc.utexas.edu/software/pylauncher/), the example slurm job script file given uses the Windows double dash that makes a long dash (–) instead of just two dashes and it resulted in an error.

-Be sure to make sure your docker container works correctly before pushing it to docker hub while in docker desktop. Docker does not use the --bind command to bind directories as Apptainer does. Instead, I needed to do the following: docker run -v /c/Users/betha/Desktop/test:/home/r-environment/output bethanyhamilton/powstudcatmod\_test:v2 R -e "source('/home/r-environment/run\_sim\_study.R')" --args PYL\_ID 3 "/home/r-environment/output". Also, you can always open up an interactive shell to test things as well (ex: docker run -it bethanyhamilton/powstudcatmod\_test:v2 /bin/bash). Finally, if your image is running, you can go to the Container tab and then navigate to the Files tab to see if your data spits out in the expected directory.

- you can use the command idev in TACC to launch an interactive shell where you can load the Apptainer module and pull your container from docker hub. Then, run the following to open up an interactive shell within the container (assuming you're in $SCRATCH): “apptainer exec /scratch/your-username/your-imagename.sif /bin/bash”. You can do a quick test of the run\_sim.R file to see if that works as expected while in the idev mode on TACC. You can also check to see if the necessary files are there and use the “cat” command to check if they are the correct versions of the files, too. I needed to do this at some point because I had made changes to the run\_sim\_study.R files and rebuilt the image with the same tag/version. I then ensured that it worked on Docker Desktop and pushed it to docker hub. However, for some reason, when I pulled it from the docker hub in TACC, it still had the old version of run\_sim\_study.R. The only way I was able to fix it was to build the image again with a new version tag, push it to docker hub, and pull the new version.