Getting Started with ARA

Brian Clark

Last Updated: May 15, 2019

Step 0: Get Familiar with the Command Line

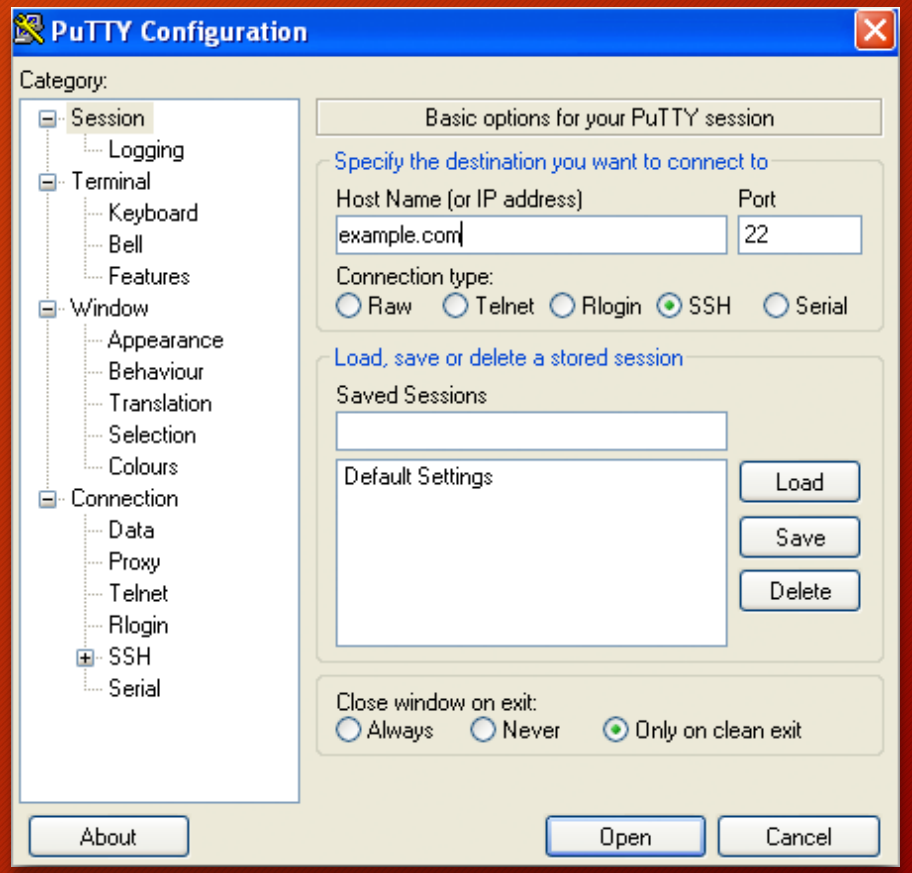
- All of our work will be done on computing clusters
- To interact with the computing cluster, we will need to use the *command line*
- It will allow us to go inside files, delete them, make new ones, run programs, etc.
- Follow this link to learn how to use it: https://www.codecademy.com/learn/learn-the-command-line

Step 1: Get an OSC Account

- For the time being, lets start by having you all do your work on the computing cluster known as ”owens”
- It is maintained by the Ohio Supercomputer Center
- Ask Amy to get an account

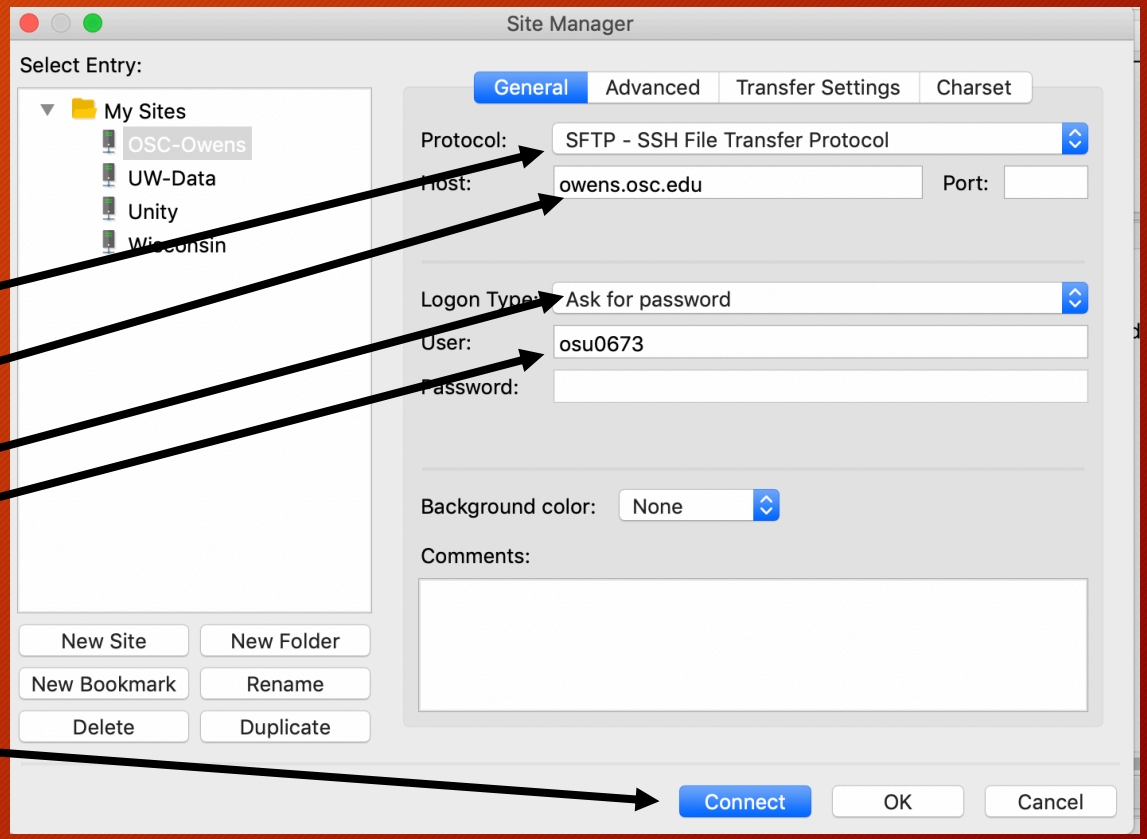
Step 2: Log-In

- Okay, now to log in!
- On a Linux or Mac:
  - Open the terminal, and type "`ssh name.#@owens.osc. edu`"
  - It might ask you to agree to an SSH security token; say yes
  - Then enter your password
- On a PC:
  - You will need to download and install an *ssh tunneling client*
  - The easiest is one called "putty": https://www.putty.org/
  - Install putty, and then open it:
    - For "Host Name", put "owens.osc. edu"
    - For Port choose "22"
    - For "Connection type" choose "SSH"
    - Click "Open"
    - You will then enter your name and password



Step 3: Learn How to Transfer Files

- Throughout your time working on any supercomputer, you'll need to transfer files
- The easiest way to transfer files is by using a file transfer client
- I recommend Filezilla: https://filezilla-project.org/
- Download and install it
- Then launch it
- Click "New Site"
  - Under "Protocol" select "SFTP"
  - Under "Host" put "owens.osc.edu"
  - Under "Logon Type" put "Ask for password"
  - Under "User" put "yourusername"
  - Then click "connect"



Step 3: Pt 2

- The way FileZilla works is very straightforward
- The left panel of your screen is *your* computer
- The right panel of the screen is the remote machine
- To transfer files you literally drag them from one screen to the next
- That's it
- Here are some nice tutorial:
  - https://www.youtube.com/watch?v=rUNQphoGVwQ
  - https://www.youtube.com/watch?v=adxmlHDim6c

Step 4: Download a copy of AraSim

- To download a copy of icemc we are going to use a piece of software called "`git`"
- This is what's known as a version control system, which allows developers to track how a piece of code evolves
- We won't use that feature for a while; for now, we're just gonna use it to access the ARA Code
- Download icemc:
  - git clone https://github.com/ara-software/AraSim

Step 5: Set Up Your Shell Environment

- Your *shell environment* defines variables you want to use
- First, put the files "`bashrc_anita.sh`" into your home directory. Yes, I know this says it's for ANITA—but I've designed it work for both!
- Open your "`.bashrc`" file and copy the contents of "`sample_bashrc.sh`" into it. Do the same with "`.bash_profile`".
- I'm having you do this because it will *source* several pieces of software you will need (like ROOT) that I've already installed for you to save time

Step 6: Compile AraSim

- Next go inside the AraSim directory by typing: cd AraSim
- Then build by typing “make”
- The make will almost certainly fail with errors that look like this

```
In file included from Primaries.cc:9:0:
counting.hh:66:23: error: ‘constexpr’ needed for in-class initialization of static data member ‘const double Counting::COSTHETAMAX’ of non-integral type [-fpe
rmissive]
   static const double COSTHETAMAX=1.0;
                       ^~~~~~~~~~~
counting.hh:67:23: error: ‘constexpr’ needed for in-class initialization of static data member ‘const double Counting::COSTHETAMIN’ of non-integral type [-fpe
rmissive]
   static const double COSTHETAMIN=0.0;
                       ^~~~~~~~~~~
counting.hh:68:23: error: ‘constexpr’ needed for in-class initialization of static data member ‘const double Counting::PHIMAX’ of non-integral type [-fpermiss
ive]
   static const double PHIMAX=2*3.14159;
                       ^~~~~~
counting.hh:69:23: error: ‘constexpr’ needed for in-class initialization of static data member ‘const double Counting::PHIMIN’ of non-integral type [-fpermiss
ive]
   static const double PHIMIN=0.;
                       ^~~~~~
counting.hh:84:23: error: ‘constexpr’ needed for in-class initialization of static data member ‘const double Counting::MIN_LOGWEIGHT’ of non-integral type [-f
permissive]
   static const double MIN_LOGWEIGHT=-3;
                       ^~~~~~~~~~~~~
counting.hh:85:23: error: ‘constexpr’ needed for in-class initialization of static data member ‘const double Counting::MAX_LOGWEIGHT’ of non-integral type [-f
permissive]
   static const double MAX_LOGWEIGHT=0;
                       ^~~~~~~~~~~~~
make: *** [Primaries.o] Error 1
```

Step 7a: Fix constexpr issue

- We need to make a change to AraSim to let it compile with C++11 and ROOT6
- Open the counting.hh file, and uncomment (remove the "//") all the lines that say "constexpr static const double" and comment out (add the "//") to all the lines without it
- Do this for all variables causing problems—there should be ~6

```
//  constexpr static const double COSTHETAMAX=1.0;
//  constexpr static const double COSTHETAMIN=0.0;
//  constexpr static const double PHIMAX=2*3.14159;
//  constexpr static const double PHIMIN=0.;

static const double COSTHETAMAX=1.0;
static const double COSTHETAMIN=0.0;
static const double PHIMAX=2*3.14159;
static const double PHIMIN=0.;
```

Uncomment these lines

Comment out these

Step 8: Test Run icemc

- Now we'll run a basic test of AraSim
- First, move the file "`test_setup.txt`" into your AraSim directory
- Now, execute the command "`./AraSim test_setup.txt`"
  - This will simualte 100 neutrinos at the energy of 10^{21} eV
- See if it executes successfully; also, you should check the outputs directory for a file called `AraOut.root`
- If it did, well done!

Step 9: Make your First Plot

- Okay, now that we have data, we can make our first plot!
- Move the following two files into your AraSim/trunk directory:
  - "`plotting_example.mk`": the makefile, which tells the computer how to put together your code
  - "`plotting_example.cc`": the code to be put together
- Run "`make -f plotting_example.mk`" (the `-f` says "use this file")
- Then, execute the plot making code: "`./plotting_example outputs/AraOut.root`"
- This will produce a plot! Revel in your brilliance.
- Put your result in our Dropbox folder, under Daily Updates, in a folder with the date and your name.
- You should actually *read* "`plotting_example.cc`" ***very carefully*** to understand everything it's doing (the explanations are in the comments)