Computing Infrastructure

- Connecting to Adroit using `ssh`
- Software environment using `module`
- Software versioning using `git`
- Compiling using `make`
- Compiling and running using MPI and slurm
- Plotting using gnuplot, IDL and python
- Bash scripting

adroit.princeton.edu

Install an X server and a terminal window on your laptop.

- for Mac OS:
  - iTerm for the terminal window
  - XQuartz for the X server

- for Windows:
  - PuTTY or KiTTY to connect remotely
  - Xming for the X server.

Get an account on adroit at https://researchcomputing.princeton.edu

Connect to adroit.princeton.edu with 2 options:

- off campus, connect first to the VPN (netID and password).
- in the terminal window, enter:
  ssh -X yourlogin@adroit.princeton.edu
  Enter your password (again…)

- off campus, connect first to the VPN.
- connect to myadroit.princeton.edu using your favorite browser.

Configuring ssh

The SSH protocol uses public key cryptography for authenticating hosts and users. The authentication keys, called SSH keys, are created using the keygen program.

On your local laptop, enter `ssh-keygen` without any arguments (just hit return).

It will create the directory `.ssh` with the private key stored in file `id_rsa` and the public key stored in file `id_rsa.pub`

Copy / paste the public key into a new file called `authorized_keys` into the remote `.ssh` directory. Make store the public key is stored using only one line (removing all line breaks).

You will not be asked for your password anymore (using `ssh` and `scp`).

Enter `ssh -X netid@adroit.princeton.edu` to allow X11 forwarding to run graphical applications remotely.

You can also edit your SSH config file. First enter the command
`cp /etc/ssh/ssh_config ~/.ssh/config`
In this new file, set the parameter `X11Forwarding yes`

Set up your adroit environment

Typically users initialize their environment when they log in by setting environment information for every application they will reference during the session.

The Environment Modules package is a tool that simplify shell initialization and lets users easily modify their environment during the session with `modulefiles`.

```
> module avail
```

```
[rt3504@adroit5 ~]$ module avail
------------------------------------------------------------------------------ /usr/share/Modules/modulefiles ------------------------------------------------------------------------------
dot  module-git  module-info  modules  null  use.own

------------------------------------------------------------------------------ /usr/local/share/Modules/modulefiles ------------------------------------------------------------------------------
boost/1.73.0      cudatoolkit/11.4             gsl/2.6                       netcdf/gcc/hdf5-1.10.6/4.7.3               netcdf/gcc/hdf5-1.10.6/openmpi-4.1.0/4.7.4
cmake/3.18.2      fftw/gcc/3.3.9               hdf5/gcc/1.10.6               netcdf/gcc/hdf5-1.10.6/4.7.4               openmpi/gcc/4.1.0
cudatoolkit/10.2  fftw/gcc/intel-mpi/3.3.9     hdf5/gcc/intel-mpi/1.10.6     netcdf/gcc/hdf5-1.10.6/intel-mpi/4.7.3     R/3.6.3
cudatoolkit/11.1  fftw/gcc/openmpi-4.1.0/2.1.5 hdf5/gcc/openmpi-4.0.4/1.10.6 netcdf/gcc/hdf5-1.10.6/intel-mpi/4.7.4     R/4.0.5
cudatoolkit/11.3  fftw/gcc/openmpi-4.1.0/3.3.9 hdf5/gcc/openmpi-4.1.0/1.10.6 netcdf/gcc/hdf5-1.10.6/openmpi-4.0.4/4.7.3 ucx/1.9.0

------------------------------------------------------------------------------ /opt/share/Modules/modulefiles ------------------------------------------------------------------------------
course/cee306/default       course/pygpu/default   intel-dpl/2021.1.2      intel-mpi/gcc/2019.7      intel-rt/2021.1.2     intel-vtune/oneapi  intel/2021.1.2
course/elecos396/default    course/pyJC/default    intel-inspector/oneapi  intel-mpi/gcc/2021.3.1    intel-tac/2020.1-024  intel/19.1(@)       java/11
course/env-geo-367/default  intel-advisor/oneapi   intel-mkl/2020.1        intel-mpi/intel/2019.7    intel-tac/2021.1.1    intel/19.1.1.217    nvhpc-nocompiler/21.5
course/pycompiled/default   intel-debugger/10.0.0  intel-mkl/2021.1.1      intel-mpi/intel/2021.3.1  intel-tbb/2021.1.1    intel/2021.1(@)     nvhpc/21.5

------------------------------------------------------------------------------ /usr/licensed/Modules/modulefiles ------------------------------------------------------------------------------
agilent/2017       ansys/AnsysEM21.1      gurobi/9.0.1  julia/0.7.0  lumerical/2020R2    matlab/R2011a  matlab/R2016a           matlab/R2020b           spark/hadoop2.7/2.2.0  stata/16.1(defau
agilent/latest     cadence/20171206       gurobi/9.1.2  julia/1.0.1  map/18.0.2          matlab/R2011b  matlab/R2016b           matlab/R2021a           spark/hadoop2.7/2.4.6  stata/17.0
anaconda3/2018.12  cadence/latest         IDL/8.4       julia/1.0.3  map/20.0.1          matlab/R2012a  matlab/R2017a           paraview/5.9.1          spark/hadoop3.2/3.2.0  synopsys/saed-mo
anaconda3/2019.3   ddt/18.0.2             IDL/8.5       julia/1.1.0  mathematica/11.0    matlab/R2013a  matlab/R2017b(default)  performance-reports/7.1 stata/11.0
anaconda3/2019.10  ddt/20.0.1             IDL/8.6       julia/1.2.0  mathematica/11.1.0  matlab/R2013b  matlab/R2018a           spark/hadoop1/1.6.1     stata/12.0
anaconda3/2020.2   EDEM/2020.0            julia/0.4.7   julia/1.3.0  mathematica/11.3.0  matlab/R2014a  matlab/R2018b           spark/hadoop2.6/1.5.2   stata/13.0
anaconda3/2020.7   gurobi/7.5.1           julia/0.5.0   julia/1.4.1  mathematica/12.1.1  matlab/R2014b  matlab/R2019a           spark/hadoop2.6/1.6.1   stata/14.0
anaconda3/2020.11  gurobi/8.0.1(default)  julia/0.5.1   julia/1.5.0  matlab/R2010a       matlab/R2015a  matlab/R2019b           spark/hadoop2.6/2.0.0   stata/15.0
anaconda3/2021.5   gurobi/8.1.1           julia/0.6.0   julia/1.6.1  matlab/R2010b       matlab/R2015b  matlab/R2020a           spark/hadoop2.6/2.1.0   stata/16.0
```

Set up your adroit environment

> module load, module list

```
[rt3504@adroit5 ~]$ module list
No Modulefiles Currently Loaded.
[rt3504@adroit5 ~]$ module load openmpi/gcc/4.1.0
[rt3504@adroit5 ~]$ module load fftw/gcc/openmpi-4.1.0/3.3.9
[rt3504@adroit5 ~]$ module load anaconda3/2021.5
(base) [rt3504@adroit5 ~]$ module list
Currently Loaded Modulefiles:
 1) openmpi/gcc/4.1.0   2) fftw/gcc/openmpi-4.1.0/3.3.9   3) anaconda3/2021.5
```

> module switch

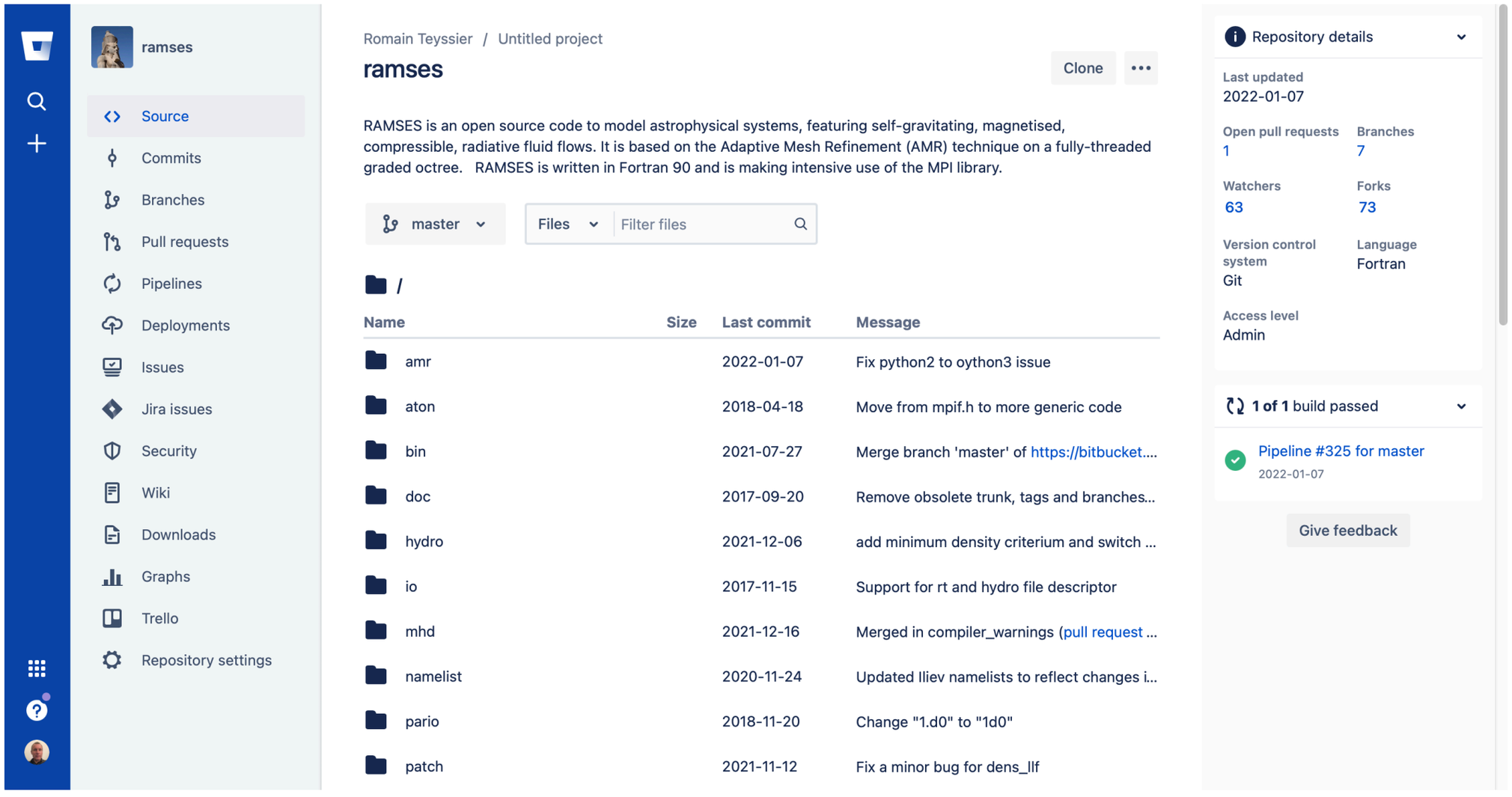
```
[rt3504@adroit5 ~]$ ifort
-bash: ifort: command not found
[rt3504@adroit5 ~]$ module load intel/2021.1
Loading intel/2021.1.2
  Loading requirement: intel-rt/2021.1.2 intel-tbb/2021.1.1 intel-mkl/2021.1.1
    intel-debugger/10.0.0 intel-dpl/2021.1.2
    /opt/intel/oneapi/compiler/2021.1.2/linux/lib/oclfpga/modulefiles/oclfpga
[rt3504@adroit5 ~]$ ifort
ifort: command line error: no files specified; for help type "ifort -help"
[rt3504@adroit5 ~]$ which ifort
/opt/intel/oneapi/compiler/2021.1.2/linux/bin/intel64/ifort
[rt3504@adroit5 ~]$ module switch intel/2021.1.2 intel/19.1
Switching from intel/2021.1.2 to intel/19.1.1.217
  Unloading useless requirement: intel-mkl/2021.1.1 intel-rt/2021.1.2
    intel-tbb/2021.1.1 intel-debugger/10.0.0 intel-dpl/2021.1.2
    /opt/intel/oneapi/compiler/2021.1.2/linux/lib/oclfpga/modulefiles/oclfpga
  Loading requirement: intel-mkl/2020.1
[rt3504@adroit5 ~]$ which ifort
/opt/intel/compilers_and_libraries_2020.1.217/linux/bin/intel64/ifort
```

Source Code Management with Git

Create your account on GitHub or BitBucket.

Git is a version control system that developers use all over the world. It helps you track different versions of your code and collaborate with other developers.

In your browser, enter: https://bitbucket.org/rteyssie/ramses



Source Code Management with Git

Clone the code to your account on adroit.

For this, in a terminal window, enter:

```
> git clone https://rteyssie@bitbucket.org/rteyssie/ramses.git
```

You have now the latest version of the RAMSES code in your HOME directory.

You can update later changes using `git pull`.

If you have write authorization, you can also upload changes using git push.

`git add foo.c`	Add a file in the local git repository.
`git rm foo.c`	Remove a file from the local git repository.
`git commit -a -m "text describing the changes"`	Commit changes to the local git repository.
`git push`	Upload all local changes to the remote repository.

Source Code Management with Git

What you can do with a Git version control system.

- track bugs to their authors
- edit users guides and wikis
- revert to older versions
- track the exact version of the code with the Git hash key
- set up an automatic build and test system
- validation and verification (reproducibility)

See examples in https://bitbucket.org/rteyssie/ramses

Compiling Code with Makefile

Makefiles are a simple way to automate code compilation.
A simple makefile contains a set of "rules" with the following syntax:

```
target: dependencies
        command1
        command2
```

A *target* is the name of a file that is generated by a program such as executable or object files. A target can also be the name of an action to carry out, such as `clean'.

A *dependency* is a file that is used as input to create the target. A target often depends on several files. Only files that have been recently modified are recompiled.

A *command* is an action that make carries out. A rule may have more than one command, each on its own line. You need to put a tab character at the beginning of every command line.

A complex example here:

```
> cd ramses/bin
> make
> make clean
```

A simple example here:

```
> cd ramses/utils/f90
> make
> make clean
```

Compiling Code with Makefile

```
[rt3504@adroit5 f90]$ more Makefile
# Makefile for RAMSES utils
BINDIR=.

# gfortran configuration
F90=gfortran
CFLAGS=-w -ffree-line-length-none -std=f2008 -Ofast -Wall -x f95-cpp-input

# ifort configuration
#F90=ifort
#CFLAGS=-cpp

LFLAGS=

MODOBJ=random.o utils.o io_ramses.o

# Make all targets
all: amr2prof amr2cylprof ramses2tipsy amr2map part2map part2prof \
     part2cube part2list partcenter part2birth part2sfr getstarlist \
     part2cylprof

amr2prof: $(MODOBJ) amr2prof.o
        $(F90) $(LFLAGS) $^ -o $(BINDIR)/$@
amr2cylprof: $(MODOBJ) amr2cylprof.o
        $(F90) $(LFLAGS) $^ -o $(BINDIR)/$@
amr2map: $(MODOBJ) amr2map.o
        $(F90) $(LFLAGS) $^ -o $(BINDIR)/$@

# Make a specific object file
%.o: %.f90
        $(F90) $(CFLAGS) -c $^ -o $@
clean:
        rm *.o *.mod
```

local variables

make all or make

targets, dependencies and actions

wild cards

phony action (clean)

Running the code and Visualize with `gnuplot`

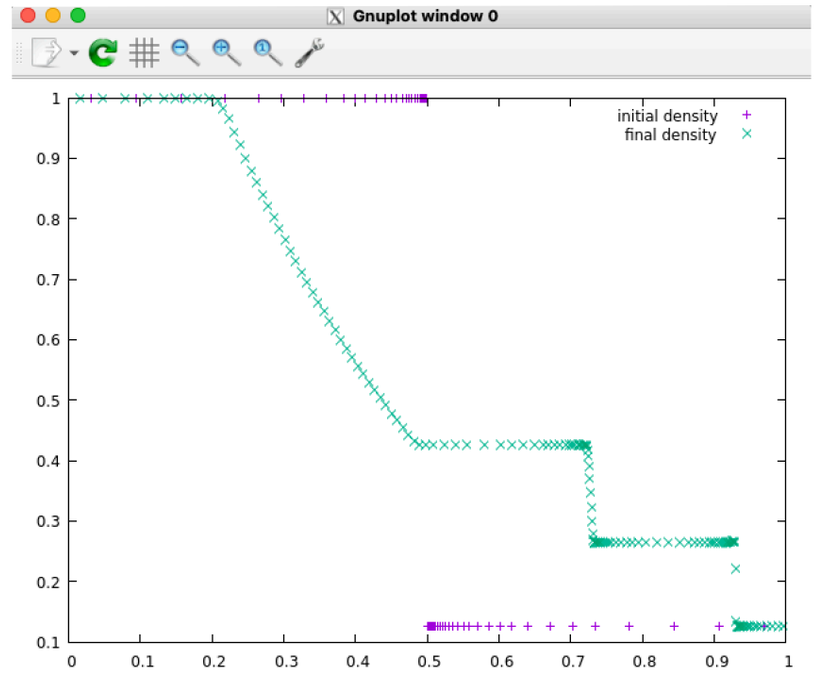
Once the code is compiled, run it with a parameter file as command line argument.

```
> bin/ramses1d namelist/tube1d.nml > run.log
```

The ket symbol > redirects the standard output to a file.
This simple example runs a standard hydro test case, the Sod shock tube.

You can visualize quickly the results using gnuplot. Enter:

```
> gnuplot
```



You are now using the gnuplot plotting program.
Make sure you have properly setup X11 forwarding and enter:

```
gnuplot> plot "run.log" every ::::1000 u 2:3 title "initial density"
gnuplot> replot "run.log" every ::1000 u 2:3 title "final density"
```

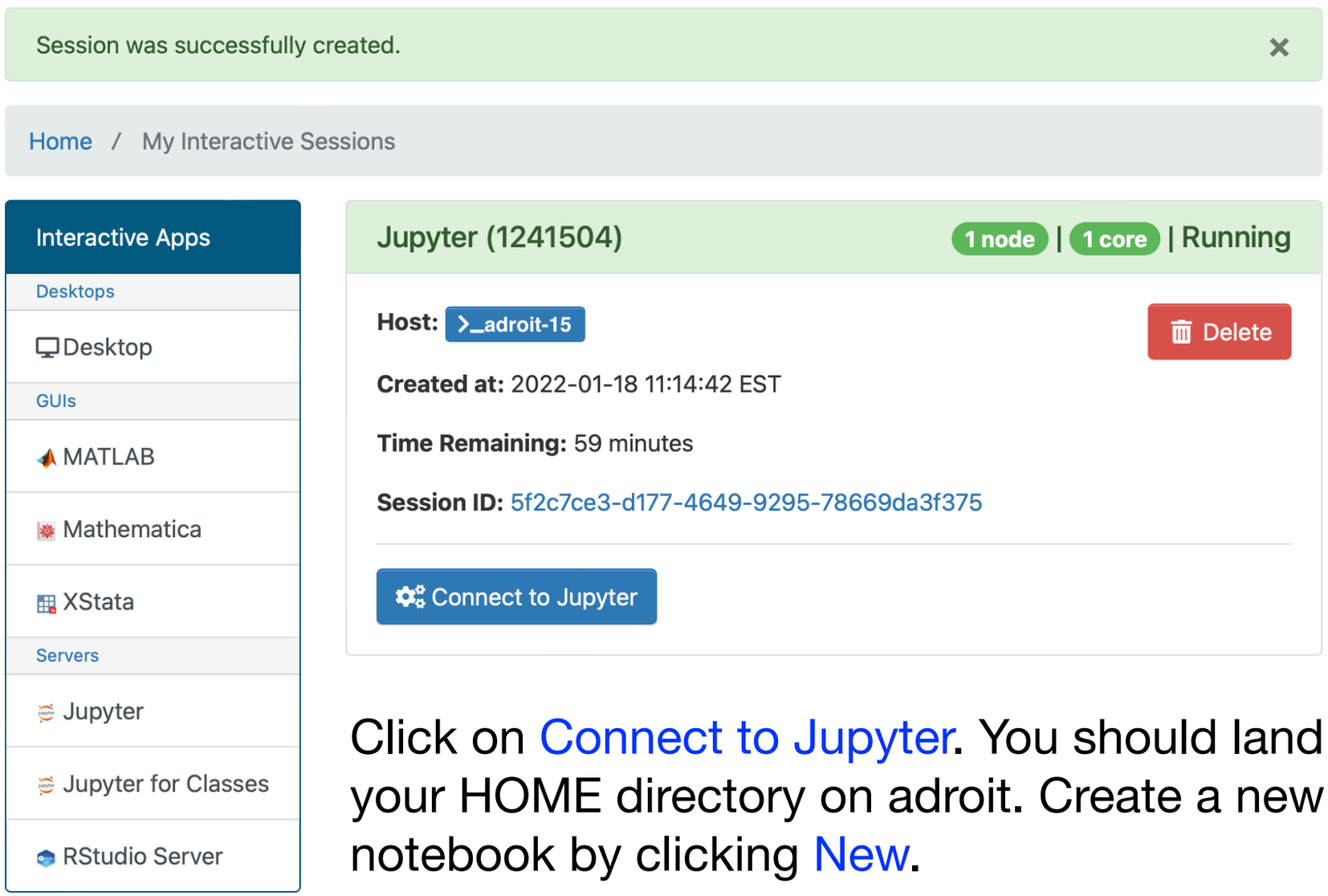
You can also use gnuplot without X11 on the command line with:

```
> gnuplot -e "set term jpeg; plot `run.log' every ::1000 u 2:3 title 'final density' " > density.jpeg
```

Visualize with a Jupyter Notebook

On your browser (after launching the VPN) connect to the web site:
https://myadroit.princeton.edu/

Launch a Jupyter Server interactive session (click Launch). After a few seconds, you should see this:



Click on Connect to Jupyter. You should land on your HOME directory on adroit. Create a new notebook by clicking New.

Visualize using jupyter notebook

Edit your own notebook following this example.

In [ ]:
```
import numpy as np
from astropy.io import ascii
import matplotlib.pyplot as plt
```

In [ ]:
```
with open("/home/yourlogin/ramses/run.log") as myFile:
    for num, line in enumerate(myFile, 1):
        if "Output" in line:
            print('line:', num, 'cells: ',line[10:14])
            i=num; n=int(line[10:14])
```

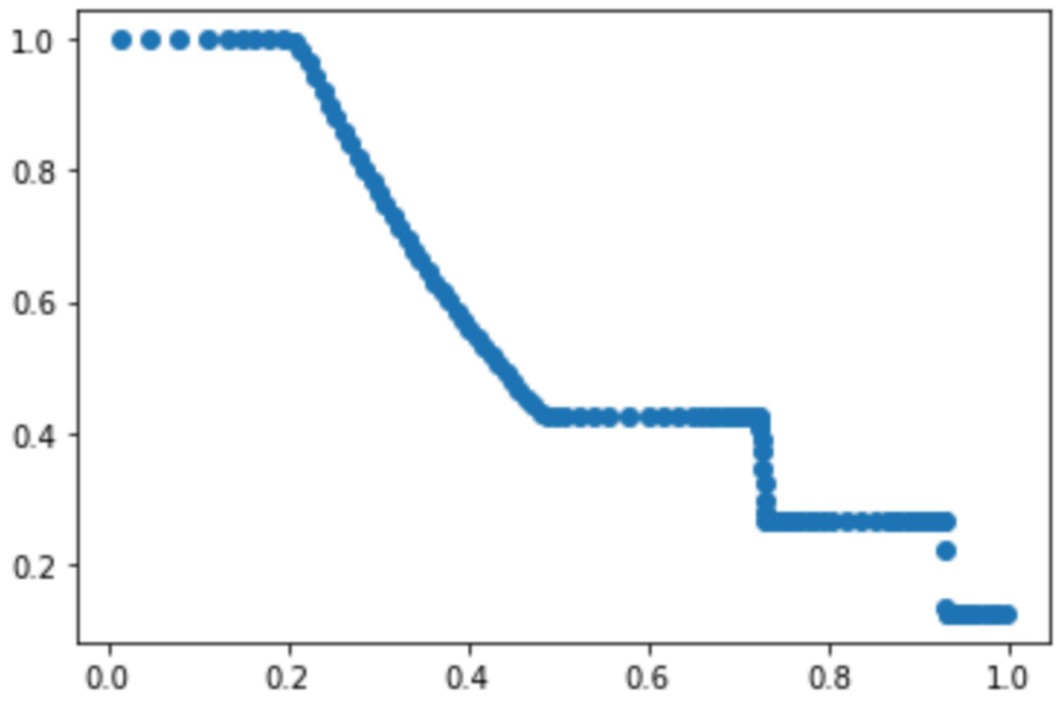
In [ ]:
```
data = ascii.read("/home/yourlogin/ramses/run.log",header_start=i-2,data_start=i-1,data_end=i-1+n)
print(data)
```

In [ ]:
```
plt.plot(data["x"],data["d"],'o')
```

If you are missing a package, you need to add it on adroit. Enter (for example) the following:

```
> module load anaconda3/2021.5
> pip3 install astropy
```

Ultimately, you should see this plot:



More details and options can be found at https://researchcomputing.princeton.edu/support/knowledge-base/jupyter

Compiling and Running with MPI

The Message Passing Library allows you to run programs in parallel on multiple processors.

```
> module load openmpi/gcc/4.1.0
> cd ramses/bin
> make clean
> make NDIM=2 MPI=1
```

Launch a parallel execution using slurm.

```
> srun -n 8 -t 00:01:00 bin/ramses2d namelist/sedov2d.nml > run.log
```

You can also edit a slurm script in a file called (for example) job.sh

```
#!/bin/bash -l
#SBATCH --job-name=sedov2d
#SBATCH --time=00:01:00
#SBATCH --nodes=1
#SBATCH --ntasks=8

module load openmpi/gcc/4.1.0
export DATE=`date +%F_%Hh%M`
srun ~/ramses/bin/ramses2d ~/ramses/namelist/sedov2d.nml > run_$DATE.log
```

Compiling and Running with MPI

Slurm allows you to submit a job on the queue

```
> sbatch job.sh
```

You can check your status on the queue

```
> squeue -u yourlogin
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           1241531       all  sedov2d   rt3504 PD       0:00      1 (Resources)
           1241532       all  sedov2d   rt3504  R       0:01      1 adroit-08
```

You can cancel your job on the queue.

```
> scancel 1241531
```

You can find more information here:
https://researchcomputing.princeton.edu/support/knowledge-base/slurm

Please use `/scratch/network/yourlogin` for the output of running jobs and to store large datasets. This space is an NFS-mounted shared space of close to 24 TB. Run the `checkquota` command to see your usage.

BASH scripting

First, run a new parallel 2D simulation with much more output files.

```
> cp namelist/sedov2d.nml sedov2d.nml
```

Modify this new file in the namelist block

```
&OUTPUT_PARAMS
tend=1.0
delta_tout=0.01
/
```

Launch a parallel execution using slurm.

```
> srun -n 8 -t 00:01:00 bin/ramses2d sedov2d.nml
```

You should see now 101 output directories.

Now edit a new bash script: `> emacs -nw movie.sh`

```
#!/bin/bash
for i in `ls -d output_00*`
do
    echo $i
    ./utils/f90/amr2map -inp $i -out dens.map -typ 1 -fil ascii -lma 8
    gnuplot -e "set term jpeg; set palette rgbformulae 22,13,-31; set pm3d map; set size square;
set cbrange [0:7]; splot 'dens.map' u 1:2:3 notitle" > pic_$i.jpeg
done
echo "converting to animated gif"
convert -delay 1 pic_output_00* movie.gif
rm -rf pic_output_00*
```

Make sure the script is executable: `> chmod a+x movie.sh`

Execute by entering: `> ./movie.sh`

BASH scripting

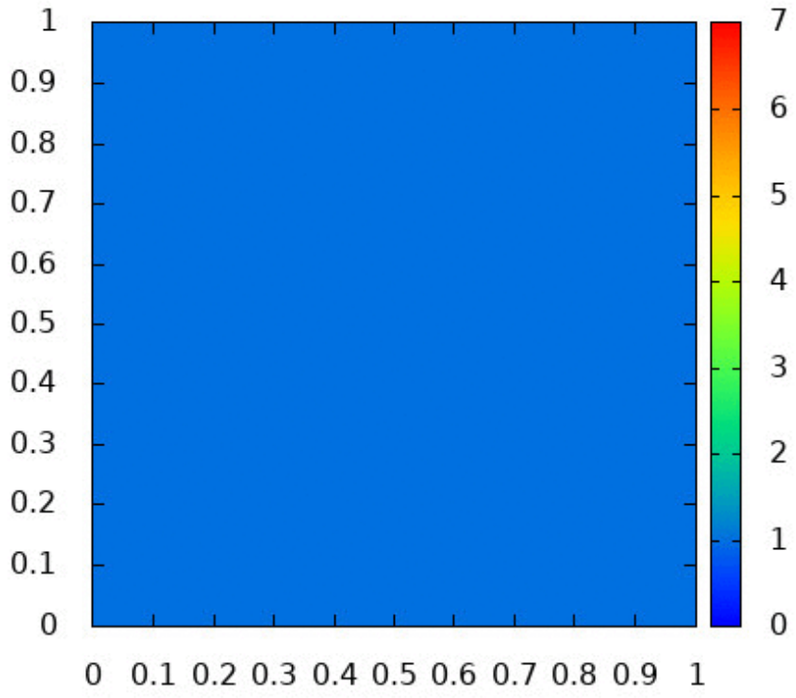
A variant for the main loop:

```
#!/bin/bash
for i in `seq -w 101`      # be careful these are backquotes or backticks
do
    echo $i
    ~/ramses/utils/f90/amr2map -inp output_00$i -out dens.map -typ 1 -fil ascii -lma 8
    gnuplot -e "set term jpeg; set palette rgbformulae 22,13,-31; set pm3d map; set size square;
set cbrange [0:7]; splot 'dens.map' u 1:2:3 notitle" > pic_00$i.jpeg
done
echo "converting to animated gif"
convert -delay 1 pic_00* movie.gif
rm -rf pic_00*
```

Copy the animated gif back to your laptop

```
> scp yourlogin@adroit.princeton.edu:ramses/movie.gif .
```

Read it with your favorite web browser. It should look like that.



Other useful Linux commands:

- `echo`: print to standard output
- `grep`: find a string in a file line by line
- `cut`: extract a string from a line
- `sort`: sort column according to some rules
- `sed`: advanced fine edition
- `awk`: advanced math operations
- `man`: manual of Linux command