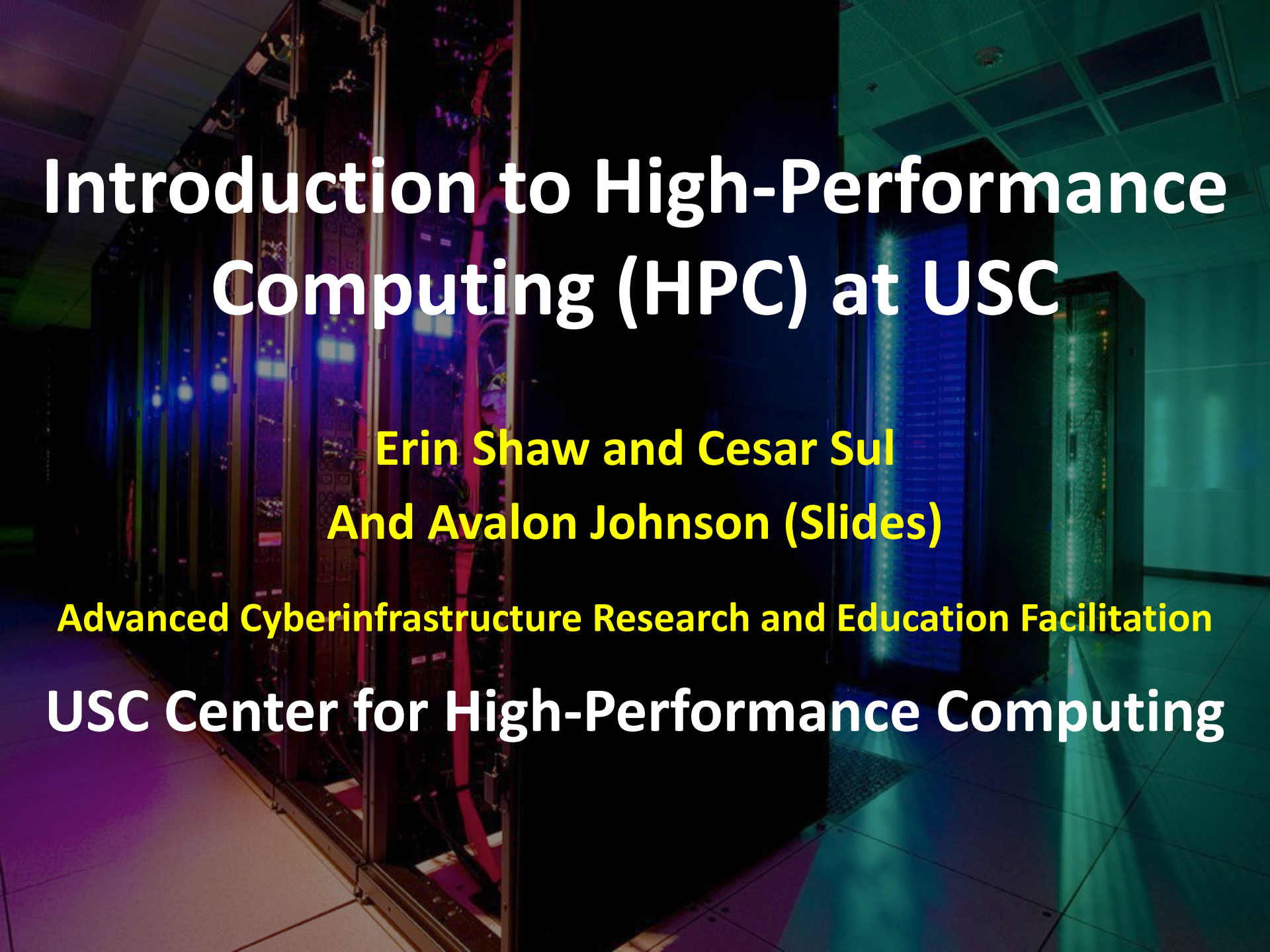


# Introduction to High-Performance Computing (HPC) at USC


Erin Shaw and Cesar Sul

And Avalon Johnson (Slides)

Advanced Cyberinfrastructure Research and Education Facilitation

USC Center for High-Performance Computing

# Welcome!

- Sign in at https://goo.gl/Qan5Kg
- Presentation handouts are available on HPC

  /home/rcf-proj/workshop/handouts

- Log in and cd to your project directory
- Make a workshop directory and copy the course materials

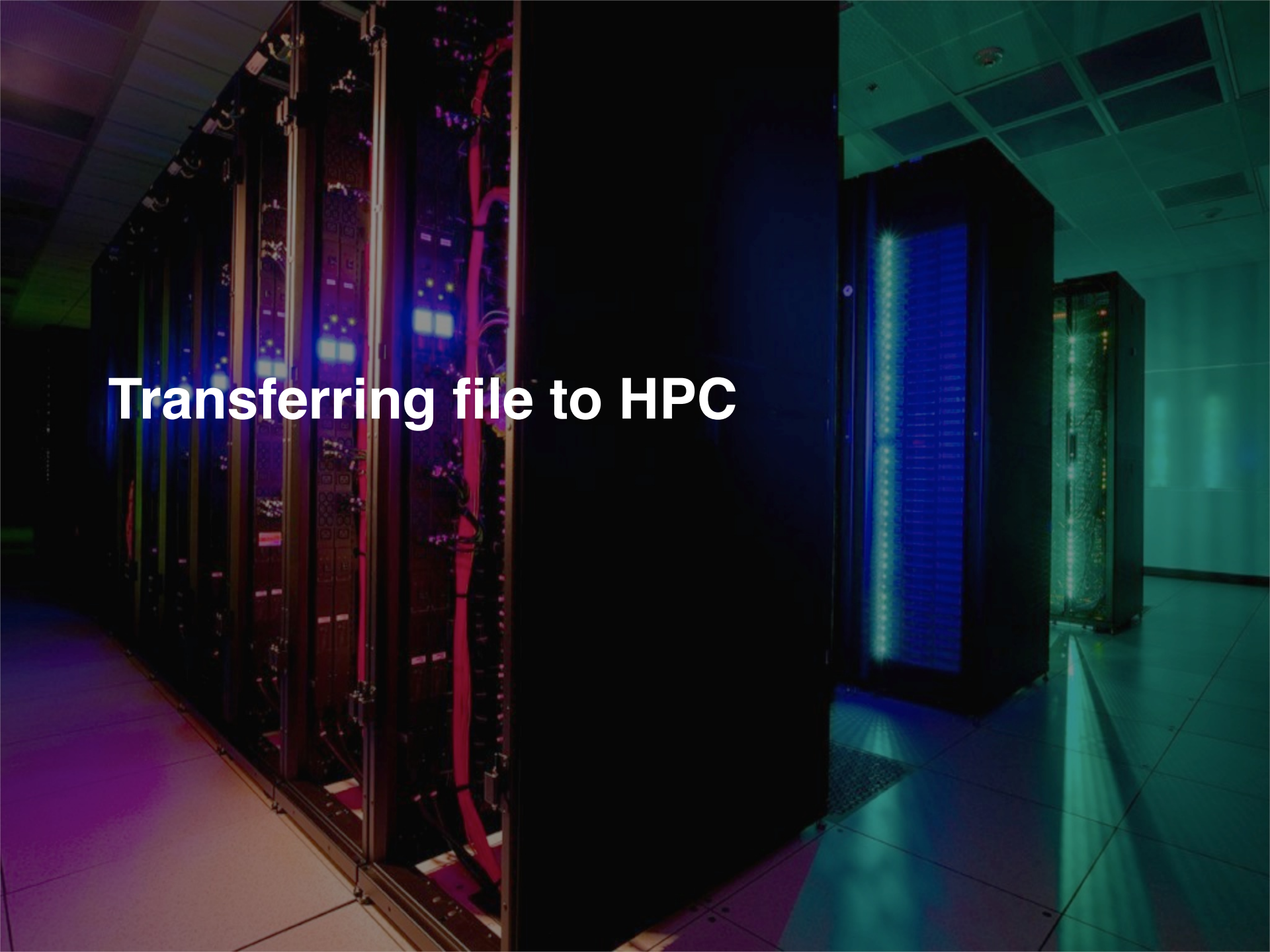
  $ mkdir workshop

  $ cd workshop

  $ cp -r /home/rcf-proj/workshop/intro-hpc/* .
- Ask questions!

# Outline

- Setup
- Running /usr/usc programs
- Running interactive jobs
- Monitoring jobs
- Running batch jobs
- Running MPI jobs
- Other slides…


Transferring file to HPC

# Transferring files

- Between HPC and Linux/macOS computers (small files)
  - scp
  - sftp (command line, FileZilla, Cyberduck,....)
- Between HPC and Linux/macOS computer (large files)
  - rsync
  - High speed transfer utilities: Globus, ascp, bbcp,...
- Between HPC and Google Drive and OneDrive
  - rclone
- From Internet to HPC
  - wget, git clone, curl

# Transferring files

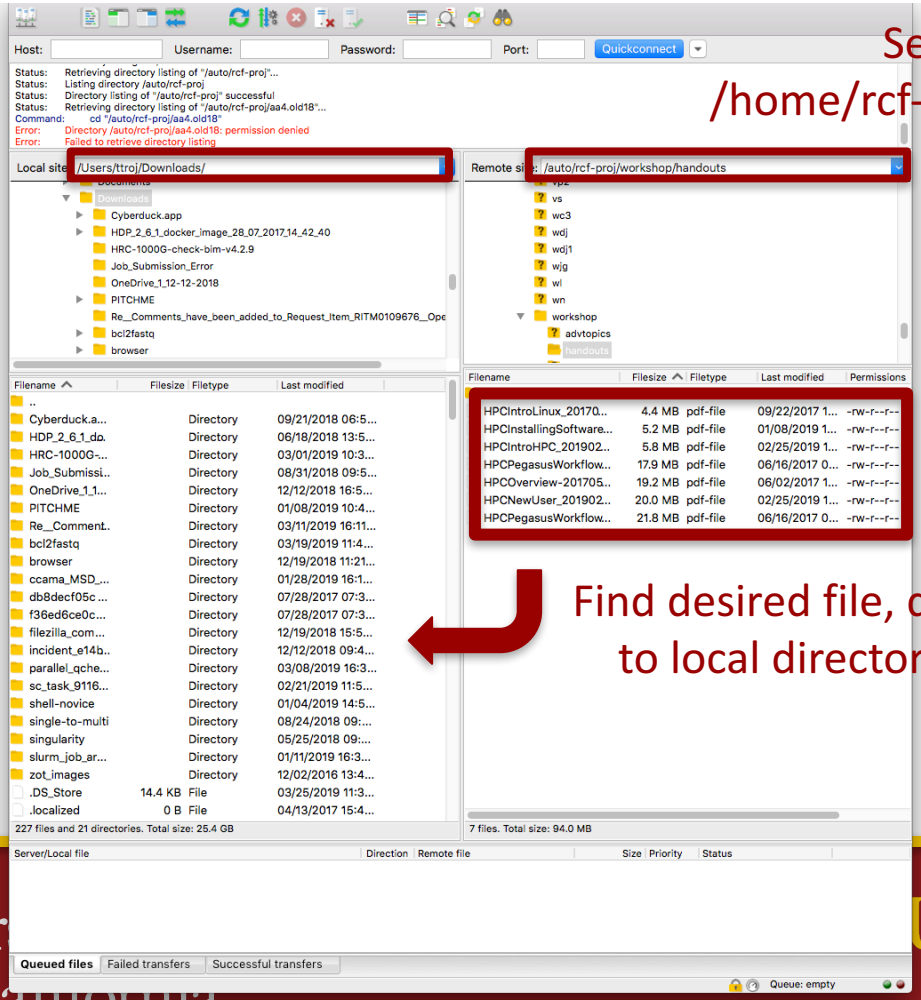
- Documentation for setting up FileZilla
  - hpcc.usc.edu -> Documentation -> Transferring Data -> Transferring Files between Your Computer and HPC
- Let's practice by downloading workshop slides to your laptop
  - /home/rcf-proj/workshops/handouts/
    - *HPCIntroHPC-yyymmdd.pdf*

# Transfering files

- Download remote file to your computer

Set local path

Set remote path to /home/rcf-proj/workshops/handou




Find desired file, drag to local directory


Running programs on HPC
- software under /usr/usc

# Software repository: /usr/usc

- HPC installs and maintains university licensed and other commonly used software in /usr/usc
  - Compilers: *gnu, intel, pgi*
  - Numerical libraries: *mpich, openmpi, cuda, fftw, petsc*
  - Molecular simulation: *NAMD, gromacs, amber*
  - Quantum chemistry: *gaussian, schrodinger*
  - Numerical computation: *MATLAB, Python*
  - Statistical computing: *R, SAS, Stata*
- You can also install software in your project directory
  - HPC can help with this

# Software repository: /usr/usc

- Let's take a field trip to the software repository

```
$ ls –F /usr/usc
```

```
acml/           cmake/          glog/           jdk/            mpich/          petsc/          schrodinger/
amber/          conf/           gnu/            julia/          mpich2/         pgi/            sgems/
antiword/       cuda/           graph-tool/     lam/            mpich-mx/       protobuf/       singularity/
aspera/         cuDNN/          gromacs/        lammps/         mvapich2/       pstotext/       sox/
bazel/          cula/           gurobi/         leveldb/        NAMD/           python/         stata/
bbcp/           dict@           hadoop/         libroadrunner/  ncview/         qchem/          subversion/
bcl2fastq/      dmtcp/          hdf5/           llvm/           netcdf/         qespresso/      swig/
bin/            etc/            hdfview@        lmod/           netcdf-fortran/ qiime/          taxila/
blast/          fdtd/           hello_usc/      lua/            nwdb/           QT/             tdk/
boost/          fftw/           hpctoolkit/     magma/          opencv/         R/              tensorflow/
caffe/          gaussian/       igraph/         mathematica/    OpenGeos/       root/           trilinos/
cellprofiler/   gcc_wrap/       imp/            matlab/         openmpi/        ruby/           udunits/
cellranger/     gflags/         intel/          modulefiles/    papi/           rust/           valgrind/
CGAL/           git/            iperf/          mongo2k/        patchelf/       samtools/       VisIt/
clang/          globus/         java@           moose/          perl/           sas/
```

# Software repository: example

- Let's run the program hello_usc

```
$ ls /usr/usc/hello_usc
1.0/ 2.0/ 3.0/ @default
$ ls -l /usr/usc/hello_usc
<listing>
$ ls /usr/usc/hello_usc/default
bin/  setup.csh  setup.sh
$ ls /usr/usc/hello_usc/default/bin
hello_usc*
```

# Software repository: example

- What happens when I run the program?

```
$ hello_usc
-bash: hello_usc: command not found
```

- Bash cannot find a program named hello_usc because it is not is your path ($PATH)

```
$ which hello_usc
```

- "which" searches $PATH
  - `echo $PATH`
  - Exercise: What happens you type `echo PATH`?

# Software repository: example

- Setup scripts add a directory to your path

```
$ source /usr/usc/hello_usc/2.0/setup.sh

$ which hello_usc
/usr/usc/hello_usc/2.0/bin/hello_usc

$ hello_usc
Hello USC!!!
I am version 2.0 running on host: hpc-login3
```

# Software repository: setup.sh

## What is in the setup script?

```
$ cat /usr/usc/hello_usc/2.0/setup.sh
1. HELLO_PREFIX=/usr/usc/hello_usc/3.0
2. PATH=${HELLO_PREFIX}/bin:$PATH
```

## What happens?

1. The variable HELLO_PREFIX is set to the location of your program (/usr/usc/hello_usc/3.0)
2. The location **${HELLO_PREFIX}/bin** is then prepended to your current path ($PATH). Bash will find the program when it searches your path.

# Software: system vs. /usr/usc programs

- Try this

```
$ which python3
/usr/bin/python3

$ source /usr/usc/python/3.6.0/setup.sh

$ which python3
/usr/usc/python/3.6.0/bin/python
```

# Software: system vs. /usr/usc programs

- Some programs and libraries (python, gcc, fftw) are installed with Linux and are updated when OS updates are installed
  - The **name** is the same as the one under /usr/usc but the programs may differ in version, libraries used, developer, etc.
  - Researchers should not use the OS version if they want reproducible results!
- Check that the software you use is correct
  - HPC creates a "default" symlink when it installs programs under /usr/usc/. The "default" version is the most recently installed and tested version of the software
  - We recommend that you replace "default" symlinks in /usr/usc/, with actual version numbers when you write your job script


Running jobs on the cluster
- Interactive jobs

# 6. Running on HPC: Easy as 1-2-3!

## 1. Plan ahead

- Organize project directory
- Install software, transfer data
- Think about your job requirements
  - amount of memory
  - number of i/o files
  - size of data

## 2. Test interactively

- Use salloc/srun
  - test run commands
  - check paths, results
- Experiment with resource allocation
  - number cores (cpus)
  - amount of memory
- Start small, then scale
- Prepare Slurm script
  - e.g., myjob.slurm

## 3. Run remotely

- Submit your job to the queue:

  $sbatch myjob.slurm
- Monitor your job to make sure it starts

  $myqueue

  $squeue --user <uname>
- Check results

  $less slurm-<jobid>.out

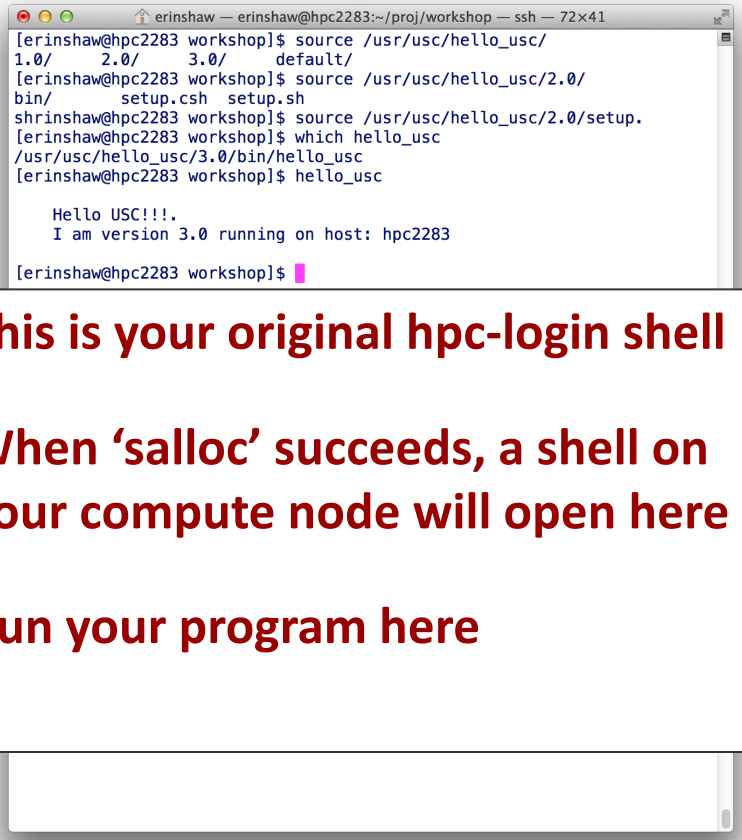
# Running a job interactively

- Let's request resources
  - cd to your workshop directory and type the following

```
$ salloc --ntasks=8 --time=00:20:00
salloc: Granted job allocation 991
salloc: Waiting for resource configuration
```

- While we're waiting for our resources…
  - open a second terminal window and log in to a head node

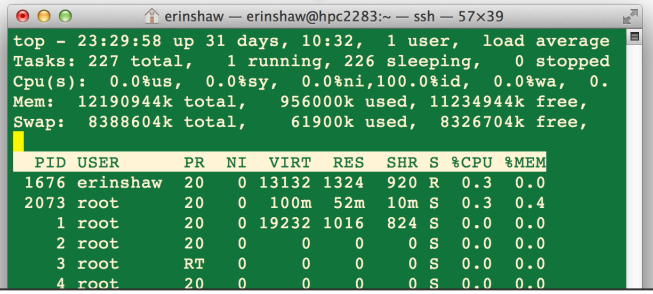
# Running a job interactively (2 windows)




This is your original hpc-login shell

When 'salloc' succeeds, a shell on your compute node will open here

Run your program here




Now open a new window and log in to head node (…ssh hpc-slurm)

Monitor your program here ($squeue or $scontrol)

You can 'ssh hpcxxx' to your compute node from here

# Got compute nodes?

```
[ttrojan@hpc-login3 ~]$ salloc --ntasks=8 --time=00:20:00
salloc: Granted job allocation 991
salloc: Waiting for resource configuration
salloc: Nodes hpc0981 are ready for job
----------------------------------------
Begin SLURM Prolog Tue 27 Feb 2018 09:49
Job ID:       991
Username:     ttrojan
Accountname:  lc_tt1
Name:         sh
Partition:    quick
Nodes:        hpc[0981,1407]
TasksPerNode: 8
CPUSPerTask:  Default[1]
TMPDIR:       /tmp/991.quick
Cluster:      uschpc
HSDA Account: false
End SLURM Prolog
----------------------------------------
```

## Slurm Prolog: Things to notice

Job ID:

***An useful number (give to us when ticketing)***

Name: sh

***Indicates an interactive job (typically job name)***

Nodes: hpc0981 hpc1407

***Shell opens on first node, run your commands here***

TMPDIR:

***This is an environment variable you can use for moving data on and off the compute node***

SCRATCHDIR: (When there are multiple nodes)

***This is an environment variable you can use for sharing files across programs on multiple nodes***

# Got compute nodes!

- We can now test our program on the cluster

```
hpc1736 $ source /usr/usc/hello_usc/3.0/setup.sh

hpc1736 $ which hello_usc
/usr/usc/hello_usc/3.0/bin/hello_usc

hpc1736 $ hello_usc
    Hello USC!!!
    I am version 3.0 running on host: hpc1736
```

# Monitoring jobs on the cluster

# Job monitoring

$ `myqueue` Display jobs status and allocated node list for your jobs

On login nodes only: Use squeue –u <me> on compute nodes

$ `squeue -u ttrojan`

| JOBID | PARTITION | NAME | USER | ST | START_TIME | NODES | SCHEDNODES | NODELIST |
|---|---|---|---|---|---|---|---|---|
| 937 | quick | sh | ttrojan | R | 2018-02-26T14:27:03 | 1 | (null) | hpc0981 |

Column **ST** displays the job status

**PD (pending)** job is queued and waiting to be executed

**R (running)** job is currently running

**CD (completed)** job has completed

# Job monitoring

```
$ squeue -j <job_id> --start
```
Displays approximate start time for <job_id

```
$ scontrol show job <job_id>
```
Display status properties of <job_id>

#Look for problems with your job

```
$ scontrol show job {jobid}
```

```
$ squeue --start --job {jobid}
```

| JOBID | PARTITION | NAME | USER | ST | START_TIME | NODES | SCHEDNODES | NODELIS |
|---|---|---|---|---|---|---|---|---|
| 937 | quick | star-lac | ttrojan | PD | 2018-02-26T14:27:03 | 2 | (null) | hpc0981 |

# Processes

- When you run a program on HPC, one or more processes are created
- You can view the processes running on a node

```
$ ps -aux
$ top
$ htop
```

(Can you find the sysadmin? ;)

- You can view node information, too.

```
$ head -5 /proc/cpuinfo
$ head -2 /proc/meminfo
```

# Linux processes

```
$top
top - 11:39:54 up  5:35, 30 users,  load average: 1.11, 0.86, 1.67
Tasks: 568 total,   1 running, 567 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.3 us,  0.8 sy,  0.0 ni, 96.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 65765680 total,  2707772 free,  6511936 used  56545972 buff/cache
KiB Swap:  8380412 total,  8297468 free,    82944 used. 58724840 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
25769 geoffret  20   0  300400  27840   4040 S  72.5  0.0   1:45.34 python
30194 geoffret  20   0  697836  33776   4352 S   8.6  0.1  14:22.32 python2.7
45495 skchoudh  20   0 1061960 368020  11924 S   5.0  0.6  16:56.82 snakemake
28078 alanjuar  20   0  151692   2660   1096 S   3.6  0.0   2:01.84 sshd
28210 alanjuar  20   0  182956   6244   4004 S   3.0  0.0   1:56.38 ssh
26361 shaoyent  20   0  119512   2932   1464 S   1.7  0.0   0:00.57 htop
35095 ztchu     20   0  151584   2488   1052 S   1.0  0.0   0:15.40 sshd
 5118 root      20   0  160880   2632   1632 S   0.7  0.0   2:16.67 top
 5702 root      20   0       0      0      0 S   0.7  0.0   2:05.83 kworker/1:2
26447 erinshaw  20   0  160752   2740   1580 R   0.7  0.0   0:00.10 top
28811 dcampo    20   0  160892   2872   1624 S   0.7  0.0   1:16.21 top
   10 root      20   0       0      0      0 S   0.3  0.0   1:04.32 rcu_sched
 3342 mdougher  20   0  160860   2872   1620 S   0.3  0.0   0:24.85 top
```

# Linux processes

```
$htop
  1  [||        8.0%]   3  [||        6.0%]   5  [|||      20.9%]   7  [|||      19.9%]
  2  [||        7.4%]   4  [||        9.3%]   6  [|||      21.6%]   8  [|||      18.8%]
  Mem[||||||||||||||||||||711M/15.5G]   Tasks: 152, 74 thr; 1 running
  Swp[|                   2.58M/8.00G]   Load average: 1.53 1.38 1.27
                                         Uptime: 05:07:43

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 3490 bk_143     20   0  149M  3668   984 S 10.7  0.0  1:03.03 sshd: bk_143@notty
 3908 scottcal   20   0  263M 12388  4244 D  5.3  0.1  2:33.76 python /home/scec-00/
 9997 georgiou   20   0  9568  1408  1168 S  4.7  0.0  0:00.41 bash /home/rcf-42/geo
 9757 joshuakl   20   0  143M  2504  1932 S  2.7  0.0  0:00.34 wget ftp://ftp.sra.eb
 3491 bk_143     20   0 71048  3196  2028 S  2.0  0.0  0:11.06 scp -r -t /staging/gr
15681 rahromos   20   0  148M  3208  1016 S  2.0  0.0  2:11.16 sshd: rahromos@notty
10229 georgiou   20   0  141M 27048  1164 S  1.3  0.2  0:06.55 /home/rcf-42/georgiou
 9646 erinshaw   20   0  118M  2516  1796 R  1.3  0.0  0:00.41 htop
17801 mdougher   20   0  156M  2736  1840 S  1.3  0.0  0:14.36 top
15682 rahromos   20   0 70680  2664  1936 S  0.7  0.0  0:58.22 /usr/libexec/openssh/
20297 root       20   0  156M  2692  1748 S  0.7  0.0  0:47.19 top
 1769 root       20   0  679M 38920  5684 S  0.7  0.2  0:10.27 /usr/bin/python /usr/
10791 root       20   0  147M  5680  4396 S  0.0  0.0  0:00.03 sshd: zlan [priv]
10019 georgiou   20   0  148M  2336   992 S  0.0  0.0  0:00.89 sshd: georgiou@pts/8
  763 root       20   0  463M 11492  6812 S  0.0  0.1  0:01.91 /usr/sbin/NetworkMana
 1784 munge      20   0  221M  2732  1560 S  0.0  0.0  0:00.64 /usr/sbin/munged --ke
27894 bk_143     20   0  147M  2172   884 S  0.0  0.0  0:00.13 sshd: bk_143@pts/0
```


Slurm Queues/Partitions

# HPC queues/partitions

- There are 4 default queues in the general (public) partition
  - A partition is automatically selected for you based on wall time and node count
  - Each partition has different constraints

| Queue Name | Maximum Wall Time | Max Node Count | Max Jobs/ User |
|---|---|---|---|
| **main** | 24 hours | 99 | 10 |
| **quick** | 2 hours | 4 | 10 |
| **large** | 24 hours | 256 | 1 |
| **long** | 336 hours | 1 | 1 |
| **largemem** (must request) | 336 hours (14 days) | 1 | 1 |
| **scavenge** (must request) | 168 hours (7 days) | 500 cores | 200 |

# HPC queues/partitions

- General queue: 'largemem'
  - `--partition=largemem`
  - Requests from pool of 4 40-core 1TB nodes
- Private queues
  - `--partition=<name>`
  - Research labs with condo nodes have private partitions with their own constraints

| Queue Name | Maximum Wall Time | Max Node Count | Max Jobs/ User |
|---|---|---|---|
| main | 24 hours | 99 | 10 |
| quick | 2 hours | 4 | 10 |
| large | 24 hours | 256 | 1 |
| long | 336 hours | 1 | 1 |
| **largemem** (must request) | 336 hours (14 days) | 1 | 1 |
| **scavenge** (must request) | 168 hours (7 days) | 500 cores | 200 |

# HPC queues/partitions

- General queue: 'scavenge'
  - `--partition=scavenge`
  - Requests idle private partition nodes
  - Short wait time, jobs may be preempted
- See compute node allocation table on website
  - https://hpcc.usc.edu/support/infrastructure/node-allocation/

| Queue Name | Maximum Wall Time | Max Node Count | Max Jobs/ User |
|---|---|---|---|
| **main** | 24 hours | 99 | 10 |
| **quick** | 2 hours | 4 | 10 |
| **large** | 24 hours | 256 | 1 |
| **long** | 336 hours | 1 | 1 |
| **largemem** (must request) | 336 hours (14 days) | 1 | 1 |
| **scavenge** (must request) | 168 hours (7 days) | 500 cores | 200 |

# Queue monitoring

- Inspect jobs in queue

```
squeue
squeue --user ttrojan (myqueue, on head nodes)
sacct -a --allocations --partition=main,large,quick,long |
    grep FAILED
```

- Inspect job status

sacct -a --allocations --partition=main,large,quick,long | grep FAILED

sacct --format=jobname%20,elapsed,maxrss%8,ntasks%4,alloctres%50,allocgres

# Queue monitoring

- Inspect node status

```
sinfo
sinfo –help (or “man sinfo” for full manual page)
sinfo --partition=main
```

```
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
main         up 1-00:00:00      3 drain* hpc[3210,3780,4659]
main         up 1-00:00:00      1  down* hpc3211
main         up 1-00:00:00      1   drng hpc4433
main         up 1-00:00:00      2  drain hpc[4434,4665]
main         up 1-00:00:00    677  alloc hpc[0965-0969,0981-1012,1015-1020,1040,11
main         up 1-00:00:00     36   idle hpc[1013-1014,1021,1041-1050,1123-1126,11
```

```
sinfo --partition=main --states=idle
sinfo --partition=main --states=idle -o"%6t %N %10G"|grep -e
gpu
```


Running jobs on the cluster
- Batch jobs

# Slurm batch jobs

- Batch jobs
  - Batch jobs are not interactive, they are submitted to Slurm and run automatically on compute nodes according to the instructions you provide in a shell script
    - *A bash script is a text file containing commands that are interpreted by bash and then run*
  - Your nodes are released when your job finishes
- Example
  - Copy the directory and files in helloUSC/ to your workshop directory

```
$ cp –r /home/rcf-proj/workshop/intro-hpc/*  .
$ cd helloUSC/
$ ls
```

# Example batch job: HelloUSC

- Try this:

```
$ nano helloUSC.slurm
```

- Edit the script
  - What is “#SBATCH --export=none”?
  - Change the path
- Submit the job and check results

```
$ sbatch helloUSC.slurm
$ squeue –user {userid}
$ ls
$ cat slurm_{jobid}.out
```

- Exercise: Edit job script so that it runs on multiple cores

# Slurm sbatch options

| Syntax | Meaning |
|---|---|
| --account=<account> | Which account to charge cpu time |
| --partition=<partition_name> | Which partition to submit to, for condo owners |
| --ntasks=<amount> | Number of CPUs to use |
| --mem_per_cpu=<amount> | RAM per CPU |
| --gres=gpu:<GPU_TYPE>:<amount> | Type and number of GPU to request |
| --time=<amount> | How long job will run |
| --constraint=<attribute> | Node property to request (avx, IB, xeon) |
| --mail-user=<email> | Where to send email alerts |
| --mail-type=<BEGIN\| END\| FAIL\| REQUEUE\| ALL> | When to send email alerts |
| --output=<out_file> | Name of output file |
| --error=<error_file> | Name of error file |
| --job-name=<job_name> | Job name |

For more details see https://slurm.schedmd.com/sbatch.html

# Slurm environment variables

- Slurm sets a number of environment variables

| Slurm | Meaning |
|---|---|
| SLURM_JOB_ID | Job ID |
| SLURM_SUBMIT_DIR | Directory job was submitted from |
| SLURM_JOB_NODELIST | File containing allocated hostnames |
| SLURM_NTASKS | Total number of cores for job |
| SLURM_SUBMIT_HOST | Hostname job was submitted from |

- See more environment variables

```
$ env | grep SLURM
```

  - https://slurm.schedmd.com/sbatch.html

# Example batch job: HelloMe.sh

- From your workshop directory

```
$ cd helloMe
```

- Run the script

```
$ ./helloMe.sh
```

- Edit the script, follow the exercise and re-run

```
$ nano helloMe.sh
$ ./helloMe.sh          #argument?
```

# Examples batch job: HelloMe.slurm

- Edit the slurm script

```
$ nano helloMe.slurm
```

- Submit the job and check results

```
$ sbatch helloMe.slurm
$ squeue --user {user}
$ ls
$ cat helloMe_{jobid}.out
```

# Requesting GPUs

- Create hellogpu.slurm and submit
- View output

```
#!/bin/bash
#SBATCH –export=none
#SBATCH --ntasks=8
#SBATCH –mem-per-cpu=1G
#SBATCH --time=00:02:00
#SBATCH –gres=gpu:2

# change to your project directory
cd /home/rcf-proj/{myproj}/{mydir}

# source setup file (setup.csh for tcsh)
source /usr/usc/intel/default/setup.sh
source /usr/usc/openmpi/1.8.8/setup.sh.intel
source /usr/usc/cuda/default/setup.sh

# run command
srun nvidia-smi
srun helloWorldMPI
```

# Running jobs on the cluster
## - Parallel jobs

# Got CPU cores?

- How many CPU cores did we run on? Assume one, unless:
  - You are running a multi-core/node application
    - *(e.g., MATLAB)*
  - Your code has multi-core/node support because you are using a library that supports parallel processing
    - *(e.g., 'parallel' for R)*
- I requested 4 cores, how do I use them?
  - "Pleasingly parallel" (formerly, "embarrassingly parallel")
    - *Little or no effort is needed to separate the problem into a number of parallel tasks. This is often the case where there is little or no dependency or need for communication between those parallel tasks, or for results between them*

```
hpc1736 $ srun hello_usc
```

# What is MPI*?

- MPI = Message Passing Interface
  - MPI is a ***specification*** for the developers and users of message passing libraries (for parallel programming).
  - MPI library ***implementations*** differ in which version and features of the MPI standard they support.
- USC HPC supports OpenMPI
  - And mpitch and mvapitch for compatibility
  - MPI will be covered in the next level HPC course

*https://computing.llnl.gov

# Using MPI interactively

## Let's compile and run an OpenMPI program

```
hpc2062: cp /home/rcf-proj/workshop/introSlurm/helloMPI/* .
hpc2062: ls
compile.sh*  helloMPI.c  helloMPI.slurm  README

hpc2062: cat compile.sh
#!/bin/sh
CC=mpicc make helloMPI

hpc2062: source /usr/usc/openmpi/1.8.8/setup.sh
hpc2062: ./compile.sh
mpicc    helloWorldMPI.c   -o helloWorldMPI

hpc2062: ls
compile.sh*  helloWorldMPI*  helloWorldMPI.c  helloMPI.slurm  README
```

# Using MPI interactively

## Results of OpenMPI program

```
hpc2062: which mpirun
/usr/usc/openmpi/1.8.8/bin/mpirun

hpc2062: srun ./helloMPI       #use srun instead of mpirun
Hello World from rank 1 running on hpc2062!
Hello World from rank 2 running on hpc2062!
Hello World from rank 3 running on hpc2062!
Hello World from rank 0 running on hpc2062!
MPI World size = 8 processes
Hello World from rank 4 running on hpc2081!
Hello World from rank 5 running on hpc2081!
Hello World from rank 6 running on hpc2081!
Hello World from rank 7 running on hpc2081!
```

# Using MPI in batch mode

- Create helloMPI.slurm and submit
- View output

```
#!/bin/bash
#SBATCH –export=none
#SBATCH --ntasks=8
#SBATCH –mem-per-cpu=1G
#SBATCH --time=00:10:00

# change to your project directory
cd /home/rcf-proj/{myproj}/{mydir}

# source setup file (setup.csh for tcsh)
source /usr/usc/openmpi/1.8.8/setup.sh

# run command
#mpirun –np 2 helloWorldMPI
srun –n2 helloWorldMPI
srun –n3 helloWorldMPI
```

# Note about node attributes myri and IB

- HPC has two clusters (interconnects)
  - Myrinet (myri) and Infiniband (IB)
  - The networks are not connected to each other
- If the interconnect attribute is not specified, HPC will use any set of nodes that allow your job to start, starting with myri nodes

  *#SBATCH --constraint=myri*

  *#SBATCH --constraint=IB*
- Warning!
  - Codes compiled to use MPICH will only run on the Myrinet nodes
  - OpenMPI codes will run on either


HPC Downtime…. coming soon!

# HPC Downtime Policy

- Twice a year HPC brings the cluster down
  - Add new machines
  - Remove old machines
  - Install patches
  - Upgrade software
  - General maintenance.
- This will never come at a good time!
  - Plan your job submission in advance -- queue will get busy!
  - Clear your data from staging when asked to do so.
  - No exceptions!

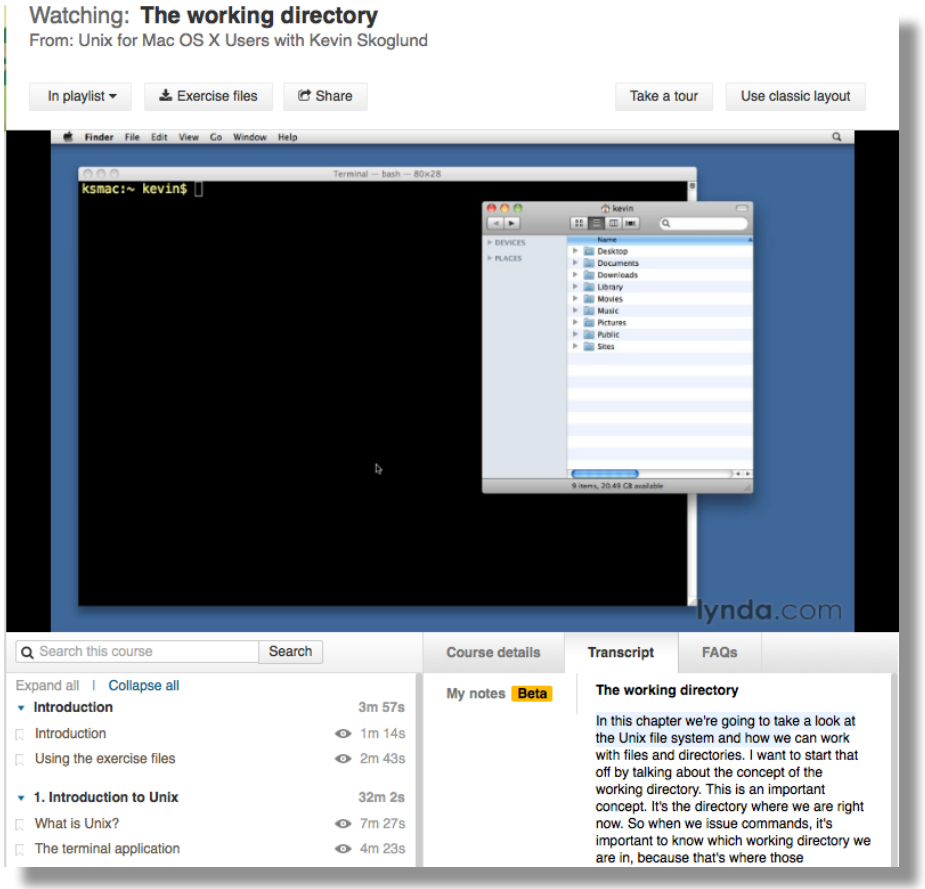
# Upgrades for downtime April 2019

- **Update the operating system on all compute nodes and head nodes.** We will apply security and operating system patches on head nodes, compute nodes, the file servers, run file system checks on the majority of the HPC file systems, and install firmware patches on various storage devices.
- **Make default symbolic link (symlink) changes for software installed under /usr/usc to newer versions.**
- **Update head node IP addresses.**
- **Add 68 HPE XL170or x86 (non-GPU) compute condo nodes to the Infiniband cluster.** The x86 nodes will have dual processor/twelve-core Intel Xeon 4116 2.1GHz CPUs, 2 terabyte (TB) internal drives, and 192 gigabytes (GB) of memory. These new nodes utilize the Intel Skylake chipsets and a 56 gigabit per second (Gbps) Infiniband connection.


Learning more

# Want to learn more?

lynda.com




- Up and Running with Bash Scripting
- Unix for Mac OS X Users
- Using Regular Expressions
- Perl 5 Essential Training
- R Statistics Essential Training
- C/C++ Essential Training
- Up and Running with Python
- Python 3 Essential Training
- Up and Running with MATLAB
- Up and Running with R

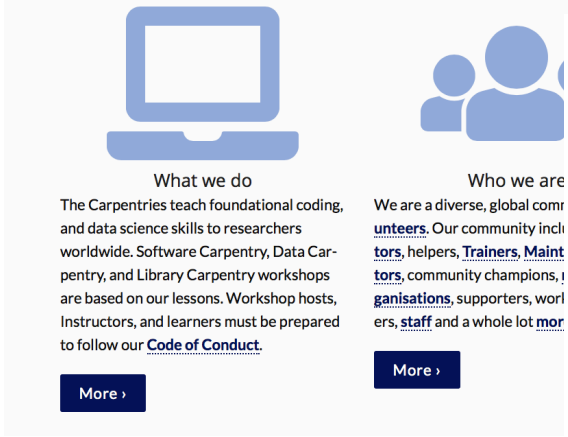
and More!

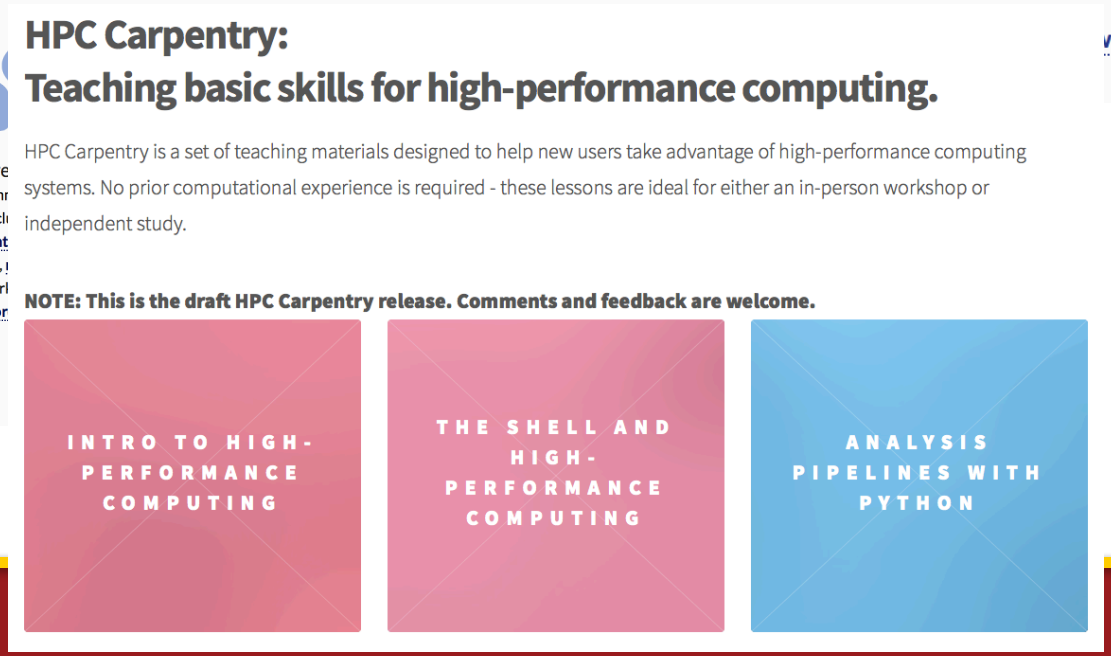
# Want to learn more?

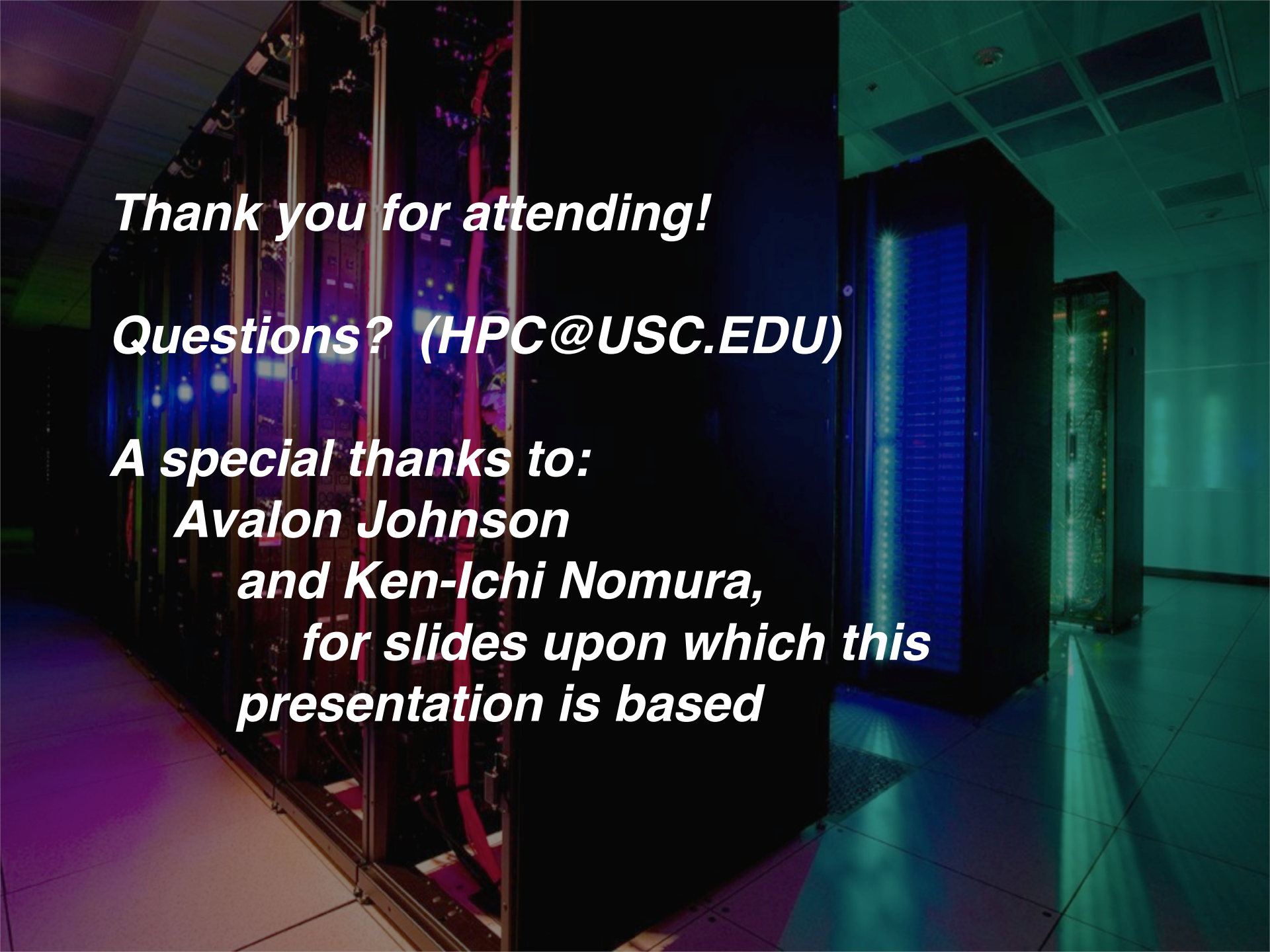
## https://carpentries.org


### Our Core Lessons in English

| Lesson | Site | Repository | Reference | Instructor Guide |
|---|---|---|---|---|
| The Unix Shell | | | | |
| Version Control with Git | | | | |
| Programming with Python | | | | |
| Plotting and Programming in Python | | | | |
| Programming with R | | | | |
| R for Reproducible Scientific Analysis | | | | |





*Thank you for attending!*

*Questions? (HPC@USC.EDU)*

*A special thanks to:*
*Avalon Johnson*
*and Ken-Ichi Nomura,*
*for slides upon which this presentation is based*