ICS-ACI Quick Reference Document

Setting ACI account

1. ICS-ACI Account Signup

- Out-of-PennState users have to apply for a PSU SLIM account first: https://ics.psu.edu/advanced-cyberinfrastructure/accounts/slim-access-accounts/
- Penn State user & SLIM account user: https://accounts.aci.ics.psu.edu/

2. 2-Factor Authentication (2FA)

- Users of the ACI system are required to have Two Factor Authentication (2FA). Users with a Penn State ID can sign up at https://2fa.psu.edu

The data directories policies

Directory	Capacity	Backup	Path
Home	10Gb	Private NFS with Backup/Recovery	/storage/home/<userid>
Work	128Gb	Shared NFS with Backup/Recovery	/storage/work/<userid>
Scratch	1 million files*	GPFS with **no Backup!** *	/gpfs/scratch/<userid>
Group	5 Tb blocks	GPFS w/ Backup/Recovery & Dual Mount option	/gpfs/group/<PI userid>/default
Archive	5 Tb blocks	Tape storage	/archive/<groupid>/default

Connect to ACI

Host Names:

- ACI (batch system, former ACI-b) **aci-b.aci.ics.psu.edu**
- ACI-i (an interactive system) **aci-i.aci.ics.psu.edu**
- ACI File Server Gateway **datamgr.aci.ics.psu.edu**

To connect to **ACI**:

- **Linux/Unix** and **Apple OS X :** standard command line **SSH** utilities

  `ssh <username>@aci-b.aci.ics.psu.edu`

  Example: `ssh abc123@aci-b.aci.ics.psu.edu`

  To enable trusted X11 forwarding use flag **–Y** (only works for ACI-b not ACI-i)

  `ssh –Y <username>@aci-b.aci.ics.psu.edu`
- **Windows : PuTTY** (free SSH client) http://www.putty.org/
- To connect to **ACI-i** please use (any OS) **: Exceed onDemand** (link to our web page)

Transferring files

- **Linux/Unix** and **Apple OS X :**
  - **scp**

To ACI: `scp –P 22 <local_file> <username>@aci-b.aci.ics.psu.edu:<target_directory>`

Example: `scp –P 22 data1 abc123@aci-b.aci.ics.psu.edu:/gpfs/work/abc123/Results`

From ACI: `scp –P 22 <username>@aci-b.aci.ics.psu.edu:<path_to_aci_file> <local_directory>`

Example: `scp –P 22 abc123@aci-b.aci.ics.psu.edu:/gpfs/work/abc123/New/data1 /local/new`

  - **rsync**

To ACI: `rsync -a ~/dir1 username@aci-b.aci.ics.psu.edu:<target_directory>`

From ACI: `rsync -a username@aci-b.aci.ics.psu.edu:<path_to_aci_file> <local_directory>`

- **Windows : WinSCP** (http://winscp.net/eng/index.php.)
  - Host Name: **datamgr.aci.ics.psu.edu**
  - File Protocol: **SFTP**
  - Port: 22
- **All OS :** FileZilla (https://filezilla-project.org/)
  - Host Name: **datamgr.aci.ics.psu.edu**
  - File Protocol: **SFTP**
  - Port: 22

Submitting a job

Non-interactive Batch Jobs

- PBS script example

```
#!/bin/bash
#PBS -l nodes=1:ppn=1          ## Requests 1 processor on 1 node
#PBS -l walltime=4:00:00       ## Requests 4 hours of walltime
#PBS -l pmem=2gb               ## Requests 2 gigabytes of memory per process
#PBS –A acb123_a_g_sc_default  ## Specifies the allocation. Use –A open for open queue
#PBS -j oe                     ## Requests that regular output and terminal output go to the same file
##  The following is the body of the script. By default PBS scripts execute in your home directory, not the
##  directory from which they were submitted. The following line places you in the directory from which the job
##  was submitted.
cd $PBS_O_WORKDIR
##  Now we want to run the program "test". "test" is in the directory that this script is being submitted from
./test >logfile
```

- To submit a pbs script to execution :

```
qsub script_name.pbs
```

Interactive Batch Jobs

- To an allocation: `qsub –I –X -A <allocName> –l [resource_list] [other_options] <script_file>`

  Example: `qsub -I –A acb123_a_g_sc_default –l nodes=1:ppn=1 -l walltime=4:00:00 script.pbs`

  where "-l" is a lower case "L"

- To open queue: `qsub –I –X -A open –l [resource_list] [other_options] <script_file>`

Checking Job Status

- *qstat* : shows the status of all your jobs
- *qstat -n* : shows the status of all your jobs and the nodes to which they were sent to
- *qstat –f <job_id>* : shows detailed information about the job *<job_id>*
- *showq –r* : shows all jobs running on ACI at the moment together with short info about them (flag *–i* shows all queued jobs)
- *showq –w acct=<alloc_name> -r* : shows jobs running within given allocation

Deleting a job

- ***qdel <job_id>*** : deletes the job identified by *job_jd*
- ***qdel –u <userid>*** :deletes all of a users jobs (you can only delete your jobs)
- ***Elevated qdel***

Using Software

Module System

Environment Modules provide a convenient way to dynamically change the users' environment through modulefiles. This includes easily adding or removing directories to the PATH environment variable.

A modulefile contains the necessary information to allow a user to run a particular application or provide access to a particular library. All of this can be done dynamically without logging out and back in. Modulefiles for applications modify the user's path to make access easy. Modulefiles for Library packages provide environment variables that specify where the library and header files can be found.

Module Families

Software is now built within families to decrease the chances of having mixed header files and libraries. The command module avail will only show the modules that are currently available within the family that you have loaded. You can use the command module spider to find the family you must load to have the software available. For example, the command *module spider software_name* will show you all available versions for the software, and *module spider*

software_name/version will show you which modules you need to load before you can load this one.

Environment Module System commands

- *module list* : lists the currently loaded modules
- *module avail* : shows modules available with currently loaded compiler and pre-compiled software
- *module spider* : lists all possible modules available on the system
    - *module spider package_name* : describes particular module and shows all available versions
    - *module spider package_name/version* : describes package particular version and shows how you can load it
- *module load package_name/version* : loads desired packages
- *module save* : you may save the list of modules which you want to be loaded during login
- *module restore* : loads the modules saved by *module save* command