The WRDS Cloud Manual

The WRDS Cloud is a high-performance computing (HPC) platform designed to be efficient, resilient, flexible, and powerful. It is a tightly integrated cluster of compute nodes managed by a centralized job engine that intelligently coordinates and oversees submitted jobs.

About the WRDS Cloud

Architecture

The WRDS Cloud is powered by the latest Dell servers with up to 24 cores and 128 GB of memory each, running Redhat Enterprise Linux managed by Univa Grid Engine, which handles grid job distribution and management. Over 350 TB of research data is stored on a highly resilient NetApp storage cluster, and is served to the grid over a high-speed 10GB Cisco fiber network. The WRDS Cloud operates over two physical data centers for redundancy, and offers high-resiliency power, cooling, and network safeguards to ensure reliable access 24/7.

Organization

The WRDS Cloud is comprised of a pair of head nodes and a grid of 25 compute nodes. When you connect to the WRDS Cloud, you are connecting to one of these head nodes -- when you submit a job to run on the WRDS Cloud, you are submitting the job to one of the available compute nodes for processing. Head nodes are designed for high-concurrency traffic, while the compute nodes are designed for CPU- and memory-intensive job execution. Both your home directory and the scratch directory are shared across all nodes, which allows you to work on any compute node in the grid without needing to transfer any data between systems. Accessing and working with these directories is described in detail in the section **Features of the WRDS Cloud**.

Data

The WRDS Cloud offers access to all WRDS data, including the NYSE Monthly TAQ (Trade & Quote) and Daily TAQ (TAQ Millisecond) Databases. Just like any other WRDS service, your institution must have an active subscription to a dataset in order to access it, though we also offer several free datasets on the WRDS Cloud should you wish to demo the service.

Access

The WRDS Cloud is open to anyone with a WRDS faculty, staff, research assistant (RA) or PhD account, and is included in your WRDS subscription. It may be accessed via SSH, PC-SAS, R/RStudio, Python, MATLAB, SAS Studio, and most popular file transfer methods. Users already familiar with Unix systems should find the WRDS Cloud familiar and easy to use, and will only need to learn how to use the Grid Engine to submit jobs, which is covered in this document. Users unfamiliar with Unix systems should consult our Unix Access Documentation (/wrds/support/Accessing%20and%20Manipulating%20the%20Data/_002Unix%20Access/index.cfm) to help with getting started, and then proceed through the rest of this document once ready.

Top

Features of the WRDS Cloud

Job Limits

The WRDS Cloud manages submitted jobs via a centralized Grid Engine process that oversees job execution in a highly efficient manner. In order to allow for optimal resource sharing for all WRDS subscribers, the following default limits apply:

- Each user may run up to **5 concurrent jobs**, and may queue any additional jobs. Additional jobs will wait in the queue until active jobs have finished.
- Each subscribing institution may run up to **20 concurrent jobs** across all users who are members of that institution, and may queue any additional jobs. Additional jobs will wait in the queue until active jobs have finished.
- Each active job may run for **one week (168 hours)** of real time (wall clock). Jobs running after 168 hours have elapsed will be automatically terminated.

Permanent File Storage

All WRDS users are given generous permanent file storage in their user home directory. Many users store their programs and uploaded datasets here, as well as logfiles, small result files, and other miscellany. Your home directory is located here:

- **/home/[group name]/[username]**
  - You may store up to **10GB** of data in this directory.
  - Files in this directory are **never deleted** by WRDS.
  - Files in this directory are **backed up to tape**, and **snapshots** are created on a regular basis (see the *File Recovery* section of this document)

Additional permanent storage is available for purchase should you require more than 10GB of personal space. Please contact WRDS Support (/wrds/support/internal_support_request.cfm) for more information.

Temporary File Storage

All WRDS subscribing institutions are given generous shared temporary storage as well. Many users output result data to this directory for staging before downloading to their local systems. The shared temporary storage directory is located here:

- **/scratch/[group name]**
  - Members of your institution may store up to **500GB** of data in this directory, shared between all members.
  - Files in your shared scratch directory are **deleted after one week (168 hours)**.
  - Files in this directory are **not backed up anywhere**, and should be downloaded shortly after being generated by your programs.

Additional temporary storage is available for purchase should your institution require more than 500GB of shared temporary space. Please contact WRDS Support (/wrds/support/internal_support_request.cfm) for more information.

SAS WORK Directory

When running SAS programs, the SAS software stores temporary job information in its WORK directory and deletes this temporary information when its job completes, as such you should not generally ever need to interface with this directory. The one notable exception is if you are running a program other than SAS that needs highly performant local disk to store temporary data (such as an R job). On the WRDS Cloud, this temporary SAS WORK directory is located here:

- **/sastemp**
  - This directory is local to each SAS node, and is therefore highly performant, but is **not synced between nodes**.

- SAS automatically uses this directory as its WORK directory, there is no need to manually specify this option in your programs.
- If you are running something other than SAS, such as an R program, you may choose to store your temporary local work data here for performance reasons.
- SAS files in this directory are **deleted immediately following job completion**, as SAS knows that this temporary data is no longer required.
- Other files (such as those manually placed here, as with the R program example above) are **deleted after one week (168 hours)**.
- Files in this directory are **not backed up anywhere**, as this is merely a temporary directory.

Please Note: Your actual SAS output files (LST file and LOG file) are written to your current working directory by default (whatever directory you originally submitted your SAS job from), or to wherever you have explicitly specified to store this data otherwise. The **/sastemp** directory is only for temporary storage of data while processing it. You should never store result data here.

WRDS Datasets

All WRDS datasets are available on the WRDS Cloud in the directory: **/wrds**. Here you can access any dataset to which your institution actively subscribes. If you are working in SAS or PC-SAS, these dataset locations are already established for you at the start of your connection via the 'autoexec.sas' file in your home directory. If you do not see a large number of libname definitions at the start of your SAS or PC-SAS session, or do not seem to be able to access a particular dataset to which you have access via its established libname, check to make sure that your autoexec.sas file is correct.

Software and Compilers

The WRDS Cloud offers a wide array of software for your research programming needs. Much of this is supported by WRDS, meaning it is kept up-to-date and its use is supported through WRDS Support (/wrds/support/internal_support_request.cfm).

The WRDS Cloud actively supports the following software:

- SAS 9.4
- PC-SAS 9.4
- R 2.15.3 and 3.1.1
- Python 2.7.6

The WRDS Cloud also provides access to some legacy software, but does not provide for updates nor support via WRDS Support:

- Fortran (f95)

- C (gcc)

Moving from the WRDS Interactive Server (wrds3)

Previous to the WRDS Cloud, WRDS offered a Solaris system for interactive SAS access, called the WRDS Interactive Server, or wrds3. This system is being phased out and users are encouraged to move to the WRDS Cloud in its stead. To ease that transition, your wrds3 home directory is available on the WRDS Cloud in the following location:

- **/wrds3home/[group name]/[username]**
  - You may access your wrds3 home directory here and transfer the contents to your WRDS Cloud home directory.
  - This directory is still subject to the quota on wrds3, which is **750MB**.
  - This directory is provided on the WRDS Cloud for convenience only, you should not store any additional data here.

This directory will be phased out a suitable amount of time after access to wrds3 is discontinued.

Top

Connecting to the WRDS Cloud

At present, there are six methods for connecting to the WRDS Cloud and accessing WRDS datasets:

1. SSH
2. PC-SAS
3. SAS Studio
4. R / RStudio
5. MATLAB
6. SFTP/SCP File Transfer

Each connection method will be briefly discussed in the following section, with links to more specific documentation for each provided in its respective section.

SSH

The WRDS Cloud is accessible via SSH, which is the most common method of working with the WRDS Cloud. Via SSH, you may establish a remote connection, and then proceed to work in the native Unix environment provided on the WRDS Cloud. This includes running both batch and interactive jobs (see the *Working on the WRDS Cloud* section), which leverage SAS, R, or Python to query WRDS data libraries and work with the results.

SSH requires the use of an SSH Client, which is a program you run on your local computer that allows you to make SSH connections to remote systems (such as the WRDS Cloud). Some common SSH clients are listed below per platform:

- **For a Mac computer:**
    - **Terminal** - is included with your computer, and can be found at */Applications/Utilities/Terminal*
    - **iTerm2** - is a free Terminal replacement offering a more robust feature set, and can be downloaded from iTerm2.com (https://www.iterm2.com/).

- **For a Windows computer:**:
    - **PuTTY** - is a multi-purpose remote connection client that offers SSH client access, and can be downloaded from putty.org (http://www.putty.org/).
    - **WinSCP** - is a file transfer utility based around SSH, SCP, and SFTP, and can be downloaded from winscp.net (http://winscp.net/eng/index.php).

Once you have downloaded and installed one of the above (if necessary), you can then initiate an SSH connection to the WRDS Cloud as follows:

- **For a Mac computer:**
    - Type: `ssh username@wrds-cloud.wharton.upenn.edu` where 'username' is your WRDS-provided username.
    - You will be prompted for your WRDS password when you connect.

- **For a Windows computer:**:
    - In the 'Host Name' field, enter: `wrds-cloud.wharton.upenn.edu`
    - When prompted from a login, enter your WRDS-provided username.
    - When prompted for a password, enter your WRDS password.

Once you have connected to the WRDS Cloud via SSH, you will be presented with a menu listing popular commands. From here, you can run SAS, R, or Python, create programs, access WRDS datasets, and store and download result data. Please refer to the section **Working on the WRDS Cloud** below for how to use the WRDS Cloud once you've connected via SSH.

You can disconnect from the WRDS Cloud at anytime by typing `logout` .

For more in-depth information on connecting via SSH, please consult the Remote Connection to WRDS Using SSH Documentation (/wrds/support/Accessing%20and%20Manipulating%20the%20Data/_002Unix%20Access/Remote%20Access%20to%20WRDS%20using%20SSH.cfm)

PC-SAS

The WRDS Cloud is also accessible via PC-SAS. PC-SAS is a SAS application that you run on your local computer, which then signs on remotely to the WRDS Cloud. It allows for a graphical frontend for SAS programming, and is capable of both working with local data and connecting to remote data. You need to have a licensed copy of PC-SAS installed on your computer (or a library / lab computer) in order to connect to WRDS using it. PC-SAS is Windows-only software.

To connect to the WRDS Cloud via PC-SAS:

1. Open PC-SAS on your Windows computer
2. Enter the following text:

```
%let wrds = wrds-cloud.wharton.upenn.edu 4016;
options comamid=TCP remote=WRDS;
signon username=_prompt_;
```

3. Provide your WRDS username and password when prompted.

Alternatively, WRDS supports using SAS-encoded passwords to enable hardcoding your username and password in the above connection code, meaning that you don't need to provide your username and password each time you connect. Please see Encoding Your WRDS Password (/wrds/support/Accessing%20and%20Manipulating%20the%20Data/_010Encoding%20your%20WRDS%20Password.cfm) for instructions on how to do this.

For more in-depth information on connecting to WRDS via PC-SAS, please consult the WRDS Cloud and PC-SAS/CONNECT Documentation (/wrds/support/Accessing%20and%20Manipulating%20the%20Data/_003PCSAS%20Connect/index.cfm).

SAS Studio

The WRDS Cloud offers access to the power of the SAS programming language and analytic engine via an easy-to-use web interface called SAS Studio. To access SAS Studio, simply point your browser to the following URL:

https://wrds-cloud.wharton.upenn.edu/SASStudio (https://wrds-cloud.wharton.upenn.edu/SASStudio)

Then provide your WRDS username and password.

Having connected, you'll find that SAS Studio offers much the same functionality as SAS and PC-SAS. Your WRDS Cloud home directory is accessible on the left-hand side under *Folders*, and the list of already-established libraries (libnames) is listed on the left under *SAS Libraries*. You can begin coding in the right-hand pane and save your programs and results to your WRDS Cloud home directory. SAS Studio supports submitting interactive SAS commands as well as running whole programs, just like PC-SAS. SAS Studio is free to all WRDS subscribers and does not require any additional SAS licensing to use.

R / RStudio

Just as you can run R directly on the WRDS Cloud once you SSH to it, you can also connect to WRDS via your local installation of R or RStudio. To do so, you'll need to install a special driver, set up your local R environment, and configure R to connect. Instructions for doing this are contained in the document Using R with WRDS (/wrds/support/Accessing%20and%20Manipulating%20the%20Data/_007R%20Programming/_001Using R with WRDS.cfm).

MATLAB

MATLAB is another remote connection mechanism for connecting to WRDS data via the WRDS Cloud. To set up your installation of MATLAB to connect to WRDS, you're need to install a special driver, and configure MATLAB to connect. Instructions for doing this are contained in the Using MATLAB with WRDS Documentation (/wrds/support/Accessing%20and%20Manipulating%20the%20Data/_009MATLAB%20Programming/index.cfm).

File Transfer

You may freely transfer any individual data / programs stored in your home directory or the scratch directory to a local system of your choosing. There are many ways to go about doing this -- here are a few:

SFTP

Point your favorite SFTP client to: `sftp://username@wrds-cloud.wharton.upenn.edu` , where 'username' is your WRDS username.
You can use the command line `sftp` command, or use an SFTP program like CyberDuck (https://cyberduck.io/?l=en) on the Mac, or PSFTP (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html) on Windows.

SCP

Once you ssh to the WRDS Cloud, use `scp` to transfer files to your local system. Type `man scp` for usage.
Alternatively, you can use a graphical tool such as WinSCP (http://winscp.net/eng/index.php) on Windows.

Should you wish to download WRDS datasets directly, WRDS has built a high-throughput fileserver at: `wrds-sftp.wharton.upenn.edu` . You may access it in the same manner as the WRDS Cloud (described above for SFTP / SCP). The credentials for wrds-sftp are the same as for the WRDS Cloud.

WRDS also maintains a Globus (https://www.globus.org/) endpoint for high-speed academic file transfers. If your institution has a Globus endpoint as well, please contact us at WRDS Support (/wrds/support/internal_support_request.cfm) to access WRDS in this manner.

Top

Working on the WRDS Cloud

When you connect to the WRDS Cloud via SSH, you're connecting to one of the WRDS Cloud head nodes: `wrds-cloud1` or `wrds-cloud2` . These head nodes are where you'll do most of your work: writing programs, examining data, looking through log and result files, etc. However, when you are ready to submit a job to the WRDS Cloud, you'll be submitting it to one of the 25 high-performance compute nodes on the grid, such as wrds-sas6 or wrds-sas30. In this manner, the head nodes can focus on high-traffic stability and resiliency, and the compute nodes can specialize in high-throughput job processing for CPU- and memory-intensive tasks.

As such, **CPU- or memory-intensive tasks are not permitted on the WRDS Cloud head nodes** (`wrds-cloud1` and `wrds-cloud2`), and must instead be run on compute nodes as described in this section.

Types of Jobs

These are two kinds of jobs on the WRDS Cloud:

1. **Interactive Jobs** - where commands are run line-by-line and the result for each command is printed as soon as you enter it.
2. **Batch Jobs** - where a whole program is submitted, runs for a length of time, and then writes returned observations to a file.

Interactive jobs are commonly used to practice with a language, submit simple, smaller queries, or to test expected outcomes before writing a larger program. Batch jobs are the most common jobs run on the WRDS Cloud, and are best for larger, multi-step programs that are expected to take hours or days to complete. Many users use an interactive job to test commands and queries to determine the correct code for a desired output, then create a larger program out of these smaller interactive commands and submit that resulting program as a batch job to the Grid Engine.

Interactive Jobs

To initiate an interactive job, type `qrsh` at the command line from one of the head nodes and you will be promptly connected to a compute node. In the following example, a user begins an interactive job with `qrsh` from the head node, and is then connected to wrds-sas6, a compute node:

```
[username@wrds-cloud1-h ~]$ qrsh
[username@wrds-sas6-h ~]$
```

Once connected in this manner, any number of programs or interpreters can be run. For example, you could:

- Start a SAS session: `sas -nodms`
- Launch an R session running R v3: `R`
- Launch an R session running R v2: `R2`
- Launch the iPython interpreter to run Python: `ipython`
- Run your favorite UNIX commands to work with your data.

To end your interactive session and return to the head node, exit whichever program you are working in (SAS, R, Python, etc.), and then enter `logout` at the command prompt.

Writing Batch Jobs

Batch jobs are the bread and butter of performing research on the WRDS Cloud. To run a batch job, you'll need to provide a 'wrapper script' to submit to the Grid Engine. A wrapper script is simply a shell script that calls the program you want to run, and optionally provides a few extra configuration parameters to the Grid Engine if desired. A minimally-simple wrapper script contains the following:

```
#!/bin/bash
#$ -cwd
sas my_program.sas
```

All wrapper scripts must include, as their first line, the declaration of the shell your program is written in, which is also called the 'shebang'. In most cases, this should be `#!/bin/bash` or `#!/bin/sh`. The second line is an optional parameter to the Grid Engine, indicating that result files should all be placed in the current working directory -- i.e. the directory from which you submitted your job. The third line calls SAS and runs your SAS program.

Here's a more complex example of a wrapper script that demonstrates some of the optional parameters you may choose to include in your wrapper script:

```
#!/bin/bash
#$ -cwd
#$ -m abe
#$ -M email@institution.edu
echo "Starting Job at `date`"
sas my_program.sas
echo "Ending Job at `date`"
```

Where:

- `#!/bin/bash` -- the shebang, required as the first line in all wrapper scripts
- `#$ -cwd` -- optional parameter that tells the Grid Engine to output all result files in your current working directory
- `#$ -m abe` -- optional parameter that instructs the Grid Engine to send an Email when the job starts and when it completes.
- `#$ -M email@institution.edu` -- required if you specified the Email option above: where to send the notification Email
- `echo` -- optional shell command to print the start time (and end time in line 7) to output
- `sas my_program.sas` -- calls SAS and runs your program

Note the use of `echo` above. Since the wrapper script is simply a shell script, you can use any standard UNIX commands within it. For example, some users chose to put a line that deletes existing logfiles of the same name as the SAS program before running, so as not to run into filename conflicts when re-running jobs multiple times.

For more information on the optional Grid Engine parameters (lines that begin with `#$`), including many more not listed above, please consult the Grid Engine Documentation (/wrds/support/Accessing%20and%20Manipulating%20the%20Data/_000Grid%20Engine%20Users%20Guide.pdf.cfm), which lists these options on page 22-24.

If instead you were submitting a wrapper script for an R program, your wrapper script would look like this:

```
#!/bin/bash
#$ -cwd
R CMD BATCH my_program.r my_program.Output
```

Where:

- `R CMD BATCH` -- calls R in batch mode
- `my_program.r` -- is the R program you have written
- `my_program.Output` -- is the filename of the output file you would like your R program to write its results to.

Please consult the document Using R with WRDS (/wrds/support/Accessing%20and%20Manipulating%20the%20Data/_007R%20Programming/_001Using R with WRDS.cfm) for more details and code examples.

Similarly, with Python:

```
#!/bin/bash
#$ -cwd
/usr/local/sas/grid/python/bin/python PyProgram.py &> PyProgram.out
```

Where:

- `/usr/local/sas/grid/python/bin/python` -- calls python in batch mode (you must supply this full path each time)
- `PyProgram.py` -- is the Python program you have written
- `PyProgram.out` -- is the filename of the output file you would like your Python program to write its results to.

Please consult the document Using Python with WRDS (/wrds/support/Accessing%20and%20Manipulating%20the%20Data/_004Python%20Programming/_001Using%20Python%20with%20WRDS.cfm) for more details and code examples.

Submitting Batch Jobs

Now that you've written a wrapper script around your program, as detailed above, you would next submit your wrapper script to the Grid Engine with the `qsub` command. Say you named the above wrapper script *my_program.sh*, you would then submit your job to the grid engine like so:

```
[username@wrds-cloud1-h ~]$ qsub my_program.sh
Your job 330000 ("my_program.sh") has been submitted
```

When you submit your program to the Grid Engine in this way, it will schedule it to run on the least-utilized compute node available on the grid, and begin processing. Once the job completes, the Grid Engine will write your output files to the location you specified in your wrapper script -- if you used the `#$ -cwd` option, it will write these files to your current working directory.

Managing Jobs

There are many ways to report on and manipulate existing jobs:

- `qstat` -- Show all current and queued jobs you have submitted
- `qstat -u \*` -- Show all current and queued jobs submitted by all users
- `qstat -j 330000` -- Show detailed information about your job #330000
- `qstat -f` -- Show all available queues and their status (d = disabled for maintenance, a = fully utilized by jobs)
- `qhost` -- Show all compute nodes, including number of processors and amount of RAM per node
- `qhost -j` -- Same as above, but also shows jobs per node
- `qdel 330000` -- Delete your job #330000

For more options on reporting on and manipulating jobs, please consult the Grid Engine Documentation (/wrds/support/Accessing%20and%20Manipulating%20the%20Data/_000Grid%20Engine%20Users%20Guide.pdf.cfm), which explores monitoring and controlling jobs on page 40-45. You can also type `man sge_intro` at the command line to see a list of all available job management commands, each of which have their own dedicated man page as well.

Job Queues & Scheduling

When you submit a job to the WRDS Cloud, be it interactive or batch, you are submitting it to an available job queue on the system. The queue you submit your job to affects the computing resources that are available to it. The WRDS Cloud offers three job queues:

- `all.q` -- The default queue: includes all compute nodes, and allows for the highest number of job slots
- `highmem.q` -- The high-memory queue: comprised entirely of hosts with 128GB RAM, but limited to 4 slots per node
- `interactive.q` -- Special interactive-only queue: all interactive jobs will automatically use this queue

To specify a job queue, you would add the `-q` flag to `qsub` like so:

- `qsub -q highmem.q my_program.sh` -- submit your program to the least-loaded highmem compute node

By default, the Grid Engine submits new jobs to the least-loaded compute node available within the queue it was submitted to. Should you find yourself needing to specify a particular compute node to run on, you can do so by appending the hostname to the queue like so:

`qsub -q highmem.q@wrds-sas12-h my_program.sh` -- submit your program to the highmem queue on wrds-sas12

`qrsh -q interactive.q@wrds-sas5-h` -- start an interactive session on wrds-sas5

However, in almost all cases, leaving compute node selection up to the Grid Engine will ensure the fastest job completion times. Also, keep in mind that the host you are specifying must actually belong to the queue you are submitting your job to in order to be processed; jobs submitted to host / queue combinations that do not exist will fail. To see which hosts are members of which queues, use `qstat -f`.

Result Files

When your job completes, the Grid Engine will write the result files from your job and mark the job as complete. In the example of a SAS job, you would see the following output:

```
[username@wrds-cloud1-h ~]$ ls -l
-rw-r--r-- 1 username institution 34710 Jun  3 14:46 my_program.log
-rw-r--r-- 1 username institution 63487 Jun  3 14:46 my_program.lst
-rw-r--r-- 1 username institution     0 Jun  3 14:46 my_program.sh.e330000
-rw-r--r-- 1 username institution    90 Jun  3 14:46 my_program.sh.o330000
```

Here we have the following result files:

- `my_program.log` -- Log file written by the SAS process that includes any errors, warnings, or notices from the SAS program you ran
- `my_program.lst` -- Result file written by the SAS process that includes all observations from executing your SAS program
- `my_program.e330000` -- File written by the Grid Engine that contains anything written to STDERR (errors) from your wrapper script
- `my_program.o330000` -- File written by the Grid Engine that contains anything written to STDOUT (output) from your wrapper script

In the case of out SAS example above, where we used the UNIX command `echo` twice in the wrapper script, the output of those `echo` commands are what we'll find in the `my_program.o330000` file. The `my_program.e330000` file is empty because there were no errors encountered when running the job.

Note: depending on job complexity and current system utilization, it may take up to two minutes for your result files to appear on the filesystem following job completion (as reported by `qstat`).

Redirecting Output

Result files may also be redirected to other locations than your current working directory if desired. For example, some users prefer to keep all Grid Engine output files for all jobs in one directory, and all SAS output files in another. You can change where the Grid Engine writes its result files by adding Grid Engine parameters (these start with `#$`) to your wrapper script, and you can change where SAS writes its result files by providing additional parameters directly to the SAS command.

In the following wrapper script, we redirect Grid Engine output to the directory `ge_logs` and errors to `ge_errs`:

```
#!/bin/bash
#$ -o ge_logs/
#$ -e ge_errs/
sas my_program.sas
```

We could further adjust this with SAS parameters to redirect the SAS result files to the `output` directory like so:

```
#!/bin/bash
#$ -o ge_logs/
#$ -e ge_errs/
sas my_program.sas -log output/my_program.log -print output/my_program.lst
```

In this way we have redirected both Grid Engine and SAS output to separate directories within our home directory. Keep in mind that if you were to attempt to run this same program twice, the execution would fail because the SAS logfiles already exist with the same name. Be sure to either remove or rename your SAS logfiles before running again, or uniqueify your logfile names each time you submit the job.

Moving from the WRDS Interactive Server (wrds3)

If you're coming from the WRDS Interactive server environment (wrds3), and are used to simply logging in and running SAS, WRDS has written a wrapper around qsub we call `qsas` that might be more convenient for you. You can use `qsas` right from a head node to start an interactive SAS session on a compute node. You can also include any command flags you would normally supply directly to `sas` such as a SAS program to enter or any additional parameters. Here's an example that runs a sas program and specifies a specific log file to write to:

```
[username@wrds-cloud1-h ~]$ qsas programs/my_sas_program.sas -log logs/my_sas_log.l
og
Your job 330000 ("my_sas_program.sas") has been submitted
```

This will submit your program directly to a compute node without you needing to worry about wrapper files or job queues. You can then check the status of this job with `qstat`.

For an interactive SAS session, you simply call qsas with the `-nodms` parameter, without providing a SAS program:

```
[username@wrds-cloud1-h ~]$ qsas -nodms
```

This will start an interactive SAS session on a compute node. When you exit the session, you will be returned to the head node automatically.

Top

File Recovery

All files in your home directory are backed up regularly via snapshots as follows:

- **Hourly** backups for the past **7 days**
- **Daily** backups for the past **4 weeks**
- **Weekly** backups for the past **2 months**

You can access these snapshots yourself by navigating to `/home/.snapshot/` and selecting a date you wish to look through.

Say you have selected the `hourly.2015-06-03_1505` snapshot to look through. Your home directory within that snapshot would then be located here: `/home/.snapshot/hourly.2015-06-03_1505/institution/username/`

Or, if you know the exact name of a file you are looking for, you can search all available snapshots using a command like the following:

```
ls -l /home/.snapshot/*/institution/username/my_file.sas
```

As demonstrated here:

```
[username@wrds-cloud1-h ~]$ ls -l /home/.snapshot/*/institution/username/my_file.sa
s
-rw------- 1 username institution 13990 Mar 18 13:41 /home/.snapshot/daily.2015-06-
03_0010/institution/username/my_file.sas
```

I can see that this file exists in the above snapshot. I can restore it to my home directory with `cp` :

```
[username@wrds-cloud1-h ~]$ cp /home/.snapshot/daily.2015-06-03_0010/institution/us
ername/my_file.sas ~/
```

If you find that you need to recover a file older than two months, it's possible -- depending on the age of the file -- that we may still have the file backed up on tape. Please contact WRDS Support (/wrds/support/internal_support_request.cfm) to request a file recovery from tape. When doing so, please supply the exact name of the file you wish to recover, as well as the date it last existed on the system.

Further Reading

Grid Engine Documentation (/wrds/support/Accessing%20and%20Manipulating%20the%20Data/_000Grid%20Engine%20Users%20Guide.pdf.cfm) -- The full Univa Grid Engine User Guide, which goes into great depth regarding the submission and management of Grid Engine jobs.

`man sge_info` -- the man page for the Grid Engine, listing all Grid Engine commands, each of which has their own individual man page as well. Enter this on the command line on the WRDS Cloud.

Using R with WRDS (/wrds/support/Accessing%20and%20Manipulating%20the%20Data/_007R%20Programming/_001Using R with WRDS.cfm) - Detailed documentation covering the use of both R and RStudio with the WRDS Cloud, complete with usage and examples.

Using Python with WRDS (/wrds/support/Accessing%20and%20Manipulating%20the%20Data/_004Python%20Programming/_001Using%20Python%20with%20WRDS.cfm) - Detailed documentation covering the use of Python with the WRDS Cloud, complete with usage and examples.

Contacting WRDS

The WRDS Cloud was designed to be powerful yet easy to use, and was implemented and documented with feedback from users in mind. If you have ideas on improving the WRDS Cloud, or recommendations regarding this documentation, please contact WRDS Support (/wrds/support/internal_support_request.cfm). We hope you find using the WRDS Cloud an enjoyable experience.

Top

Table of Contents

(http://www.wharton.upenn.edu)

About WRDS (https://wrds-web.wharton.upenn.edu/wrds/about/index.cfm)
WRDS FAQs (https://wrds-web.wharton.upenn.edu/wrds/about/WRDS FAQs.cfm)
WRDS News (https://wrds-web.wharton.upenn.edu/wrds/news/index.cfm)

3 Ways to use WRDS (https://wrds-web.wharton.upenn.edu/wrds/about/three_ways_to_use_WRDS.cfm)
Account Types on WRDS (https://wrds-web.wharton.upenn.edu/wrds/about/Account Types.cfm)
Terms of Use (https://wrds-web.wharton.upenn.edu/wrds/about/terms.cfm)

Account Preferences (https://wrds-web.wharton.upenn.edu/wrds/mywrds/preferences.cfm)
Info / Support Request (https://wrds-web.wharton.upenn.edu/wrds/about/external_support_request.cfm)
Privacy Policy (https://wrds-web.wharton.upenn.edu/wrds/about/privacy.cfm)

WRDS Demo (https://wrds-web.wharton.upenn.edu/wrds/demo/)
Conference Calendar (https://wrds-web.wharton.upenn.edu/wrds/about/Conference Calendar.cfm)
Best Paper Awards (http://www.whartonwrds.com/best-paper-award-winners/)

Wharton Research Data Services