Table of Contents

LOGGING INTO ARC

You can connect to Arc using your abc123 id and an SSH client/terminal. Two-factor authentication is required and hence, you will need to register your account with DUO at passphrase.utsa.edu.

The hostname for accessing Arc is: **arc.utsa.edu**
The port number to connect is: 22

The complete SSH command is as follows, and `abc123` in this command should be replaced with your actual `abc123` id:

ssh -p 22 abc123@arc.utsa.edu

When you connect you will be prompted for your choice of DUO authentication mechanisms as shown below. Enter the number for the option you prefer. Follow the directions that you receive on your mobile device, and you will then be connected to Arc.

```
|Duo two-factor login for abc123
|
|Enter a passcode or select one of the following
|options:
|
| 1. Duo Push to XXX-XXX-1234
```

```
| 2. Phone call to XXX-XXX-1234
| 3. SMS passcodes to XXX-XXX-1234
|
| Passcode or option (1-3):
```

ACCESSING COMPUTE NODES ON ARC

This section covers an overview of the steps for requesting access to the compute nodes for running jobs interactively and in batch mode using the Slurm job-scheduler and workload manager. Slurm enables submitting, monitoring, and cancelling jobs on Arc.

There are multiple partitions or queues on Arc. Table 1 shows the queue-names, and the constraints on using them.

Table 1. Slurm job queues on Arc

Queue Name	Node Type	Max Nodes Per Job	Max Duration Per Job	Max jobs in Queue	Max Number of Cores per Node	Local Scratch Disk Space
bigmem	Intel Cascade-Lake	2	72 hours	10	80 physical cores 160 hyperthreads or virtual cores	1.5TB local scratch storage
compute1	Intel Cascade-Lake	20	72 hours	10	40 physical cores	1.5TB local scratch storage

					80 hyperthreads or virtual cores	
compute2	Intel Cascade-Lake	20	72 hours	10	40 physical cores 80 hyperthreads or virtual cores	No local scratch space
computedev	Intel Cascade-Lake	2	2 hours	10	40 physical cores 80 hyperthreads or virtual cores	No local scratch space
gpu1v100	One NVIDIA v100 GPU attached to Intel Cascade-Lake processor	1	72 hours	10	40 physical cores 80 hyperthreads or virtual cores	1.5TB local scratch storage
gpudev	One NVIDIA v100 GPU attached to Intel Cascade-Lake processor	1	2 hours	10	40 physical cores 80 hyperthreads or virtual cores	1.5TB local scratch storage
gpu2v100	Two NVIDIA V100 GPUs attached to an Intel Cascade-Lake processor	1	72 hours	10	40 physical cores 80 hyperthreads or virtual cores	1.5TB local scratch storage

RUNNING JOBS INTERACTIVELY WITH THE SRUN COMMAND

You can get interactive access to a compute or GPU node by using the `srun` command from a login node.

If you are proceeding with running the commands below for the interactive session, please skip the steps for the batch job submission corresponding to each example.

Run the following command to get interactive access on a non-GPU compute node:

```
srun -p compute1 -n 1 -t 01:30:00 --pty bash
```

Run the following command to get interactive access on a GPU node:

```
srun -p gpu1v100 -n 1 -t 01:30:00 --pty bash
```

Note: There are multiple GPU and non-GPU partitions/queues on Arc. Please substitute the right partition/queue name (e.g., compute1 and gpu1v100) in the `srun` commands above.

SUBMITTING BATCH JOBS WITH THE SBATCH COMMAND

The Slurm sbatch command is used to submit a batch job to one of the Arc queues mentioned in Table 1 and following is its syntax:

```
[abc123@login003 ~]$ sbatch job_script.slurm
```

In the command above, **`job_script.slurm`** is a text file containing `#SBATCH` directives and shell commands that describe how the job that is being submitted will run. The details of the job script's contents depend on the type of job that you wish to run.

In the job script :(1) use `#SBATCH` directives to request computing resources (*e.g.,* 2 nodes for 1 hour); and then (2) use regular Linux shell commands to specify the tasks that you would like to do once your job starts running. You can choose to run a single executable (of an application), or can combine different executables in the same job-script to run either one after another or simultaneously.

Table 2. Options to use with the `#SBATCH` directive in a Slurm job-script

Options	Full form of Option Name	Description
`-J`	--job-name	· Specify a name for the job · This is a required option
`-o`	--output	· Instruct Slurm to write the job's standard output to the file named after the -o flag · This is a required option
`-p`	--partition	· Request a specific partition or queue for running the job · This is a required option
`-t`	--time	· Specify the maximum anticipated run-time of the job · This is a required option
`-N`	--nodes	· Specify the number of nodes required for the job · This is a required option

-n	--ntasks	· The number of tasks that will be launched on a node · The maximum number of tasks can be set as equal to the maximum number of physical cores in the node in a queue, or the maximum number of hyperthreads or virtual cores · This is a required option
--mail-user=email_id		· Replace email_id with your actual email id to receive notifications related to your job start, end, and failure. This option is meant to be used in conjunction with the **–mail-type** option · Using this option is not required
--mail-type=begin		· Here, a mail will be sent to notify the beginning of the job · "begin" can be replaced with "end", "fail", or "all" to receive other notifications · This option is meant to be used in conjunction with the **–mail-user option** · Using this option is not required

A sample batch job-script is shown in Listing 1 and the options are explained. Make sure you are connected to the Arc system (`ssh -p 22 username@arc.utsa.edu`) and are on the login node for submitting a batch job.

```
#!/bin/bash
#-----------------------------------------------------
# Sample Slurm job script for Arc nodes


#SBATCH -J myjob              # Job Name
#SBATCH -o myjob.o%j          # Name of the stdout output file
#SBATCH -e myjob.e%j          # Name of stderr error file
#SBATCH -p compute1           # Queue (partition) name
#SBATCH -N 1                  # Total # of nodes (must be 1 for serial)
#SBATCH -n 1                  # Total # of mpi tasks (should be 1 for
serial)
#SBATCH -t 01:30:00           # Run time (hh:mm:ss)


#Put other commands such as module load if needed after this line (All
commands must follow #SBATCH directives)
# Launch code...


./mycode.exe
```

Listing 1. Sample Slurm Job-Script

Note-1: If you want to activate the email notifications for tise job then you will need to add the following email in the Slurm job script shown above before you list any of the Linux shell commands to run the job.

```
#SBATCH --mail-type=ALL
#SBATCH --mail-user=username@utsa.edu
```

Note-2: The following Slurm job options are mandatory and must be included with every job: `*[-n,-N,-t,-p]*`

OTHER SLURM COMMANDS FOR JOB MANAGEMENT

After you have submitted a job through Slurm - either interactively or through the batch job-script - there are additional commands that you can run to monitor or cancel the job, or create dependencies such as holding the execution of a future job till the current one has completed.

Table 3 : Overview of Other Job Management commands

Command	Syntax	Example
`squeue`	`squeue -u uname`	`squeue -u abc123` Here `abc123` is the username with which you log into Arc.
`scancel`	`scancel jobid`	`scancel 8088547` Here `8088547` is the job id that you received after your job was submitted.
`sinfo`	`sinfo`	`sinfo`
`sbatch --dependency`	`sbatch --dependency=afterok:jobid jobscript`	`sbatch --dependency=afterok:8088568 new_job.slurm` Here `8088568` is the job id that you received after you submitted a previous job on which you would like to create a dependency in the current job.
`scontrol`	`scontrol show job=jobid`	`scontrol show job=8090259`

		Here `8090259` is the job id that you received after you submitted a previous job, whose detailed information you wanted to check
`sacct`	`sacct --starttime date`	`sacct --starttime 2021-07-10` Here `2021-07-10` is the date on ar after which you want to see the jobs

SQUEUE

You can check the status of your job - whether it is pending, running, or completing - by using the squeue command. This command can also help you find out the job ids associated with all your pending or running jobs so that you can use it with the `scancel` or `sbatch --dependency` command (its syntax is also shown in Table 3).

If you just enter the `squeue` command and press enter, it will show you the list of all the jobs currently waiting or running on the system (and not necessarily yours):

```
$ squeue
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
1429  gpu2v100 test_job   abc123 PD       0:00      2 (PartitionNodeLimit)
```

If you specify your username with the -u option to the `squeue` command then it will show you the status of all the jobs associated with your account:

```
$ squeue -u abc123
```

A sample output of the squeue command looks as follows:

```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
 1429  gpu2v100 test_job   abc123 PD       0:00      2
(PartitionNodeLimit)
```

The column labeled "ST" displays each job's status:

- "PD" means "Pending" (waiting)
- "R" means "Running"
- "CG" means "Completing" (or cleaning up after exiting the job script)

If you specify the `--start` option to the `squeue` command then it will show you the status of all the jobs associated with your account:

```
$ squeue --start
             JOBID PARTITION     NAME     USER ST          START_TIME  NODES
SCHEDNODES           NODELIST(REASON)
              1429  gpu2v100 test_job   abc123 PD                 N/A      2
(null)                (PartitionNodeLimit)
```

SCANCEL

You can cancel a job as long as it is in the pending or running state using the `scancel` command. A sample of `scancel` command is shown in Table 3. You can find out the job ids associated with all your pending or running jobs by using the `squeue` command (its syntax is also shown in Table 3).

```
$ squeue -u abc123
             JOBID  PARTITION       NAME       USER  ST          TIME   NODES
NODELIST(REASON)
              1429   gpu2v100  test_job    abc123  PD          0:00        2
(PartitionNodeLimit)
              1440  compute1     bash    abc123  R       4:11      1 c001


$ scancel 1440
```

SBATCH --DEPENDENCY

The `sbatch` command can be used to manage workflows that involve multiple steps such that the output of one step (or job) is the input to another step (or job) by using the

`--dependency` option. The syntax and example of using this command is shown in Table 3. As an example, let us assume that we submit our first job (`new_job1.slurm`) as below and get a job id `1441` by the Slurm job scheduler.

```
$ sbatch new_job1.slurm

Submitted batch job 1441
```

If the completion of this job is required before we submit another job (`new_job2.slurm`), then we can create a dependency between this new job and the old job (`1441`) as follows:

```
$ sbatch --dependency=afterok:1441 new_job2.slurm

Submitted batch job 1442
```

SINFO

You can monitor the status of the different partitions or queues on Arc using the `sinfo` command. As an example, the following command will show the number of nodes that are Available, Idle, Other, and Total in each partition on Arc:

`sinfo -S+P -o "%18P %8a %20F"`

```
$ sinfo -S+P -o "%18P %8a %20F"
PARTITION          AVAIL    NODES(A/I/O/T)
bigmem             up       0/2/0/2
compute1*          up       0/65/0/65
compute2           up       0/25/0/25
compute3           up       0/0/0/0
computedev         up       0/5/0/5
gpu1v100           up       0/30/0/30
gpu1vector         up       0/0/0/0
gpu2a100           up       0/0/0/0
gpu2v100           up       0/5/0/5
gpudev             up       0/2/0/2
softmatter         up       0/20/0/20
testing            up       0/3/1/4
```

SCONTROL

The `scontrol` command provides detailed information about the configuration of a specific job. It is used to view or modify Slurm configuration including: job, job step, node, partition, reservation, and overall system configuration. As an example, let us assume that we have submitted a job (new_job1.slurm) as below and get a job id `1441` by the Slurm job scheduler.

```
$ sbatch new_job1.slurm

Submitted batch job 1441
```

We can see the detailed information about this job as follows:

```
$ scontrol show job=1441
JobId=1441 JobName=helloWorldC
   UserId=abc123(860633341) GroupId=abc123(860633341) MCS_label=N/A
   Priority=20681 Nice=0 Account=abc123 QOS=normal
   JobState=COMPLETED Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:00 TimeLimit=00:02:00 TimeMin=N/A
   SubmitTime=2021-07-12T20:29:38 EligibleTime=2021-07-12T20:29:38
   AccrueTime=2021-07-12T20:29:38
   StartTime=2021-07-12T20:29:38 EndTime=2021-07-12T20:29:38 Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   LastSchedEval=2021-07-12T20:29:38
   Partition=compute1 AllocNode:Sid=c001:19615
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=c002
   BatchHost=c002
   NumNodes=1 NumCPUs=80 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=80,node=1,billing=80
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=NO Contiguous=0 Licenses=(null) Network=(null)

Command=/home/abc123/documentation/serial_jobs/serialc/GNU/helloworldc.slu
rm
   WorkDir=/home/abc123/documentation/serial_jobs/serialc/GNU

StdErr=/home/abc123/documentation/serial_jobs/serialc/GNU/helloWorldC.txt
   StdIn=/dev/null

StdOut=/home/abc123/documentation/serial_jobs/serialc/GNU/helloWorldC.txt
   Power=
```

SACCT

The `sacct` command displays job accounting data stored in the job accounting log file or Slurm database in a variety of forms for your analysis. This command displays information on jobs, job steps, status, and exit codes by default. You can tailor the output with the use of the --format= option to specify the fields to be shown. As an example, following command will show all the jobs that started on or after the date passed as an argument to the starttime option:

```
$ sacct --starttime 2021-07-09
       JobID    JobName  Partition    Account  AllocCPUS      State
ExitCode
------------ ---------- ---------- ---------- ---------- ---------- ------
--
1429           test_job   gpu2v100     abc123          4    PENDING
0:0
1432         helloWorl+   compute1     abc123         80  COMPLETED
0:0
1432.batch        batch                abc123         80  COMPLETED
0:0
1432.extern      extern                abc123         80  COMPLETED
0:0
1437         helloWorl+   compute1     abc123         80     FAILED
11:0
1437.batch        batch                abc123         80     FAILED
11:0
1437.extern      extern                abc123         80  COMPLETED
0:0
```

RUNNING JOBS ON THE COMPUTE NODES ON ARC

A sample R script is shown below and all this code does is print "Hello World!!" to standard output.

```
print("Hello World!!")
```

Add execute permissions on the R script file as follows:

```
[username@login001]$ chmod +x program_name.r
```

The R program can be run either in batch mode using a Slurm batch job-script or interactively on a compute node.

Running the script in Interactive-Mode**: The executable can be run interactively on a compute node using the following set of commands and the output will be displayed on the terminal:**

```
[username@login001]$ srun -p compute1 -n 1 -t 00:05:00 --pty bash

[username@c001]$ ml gnu8/8.3.0

[username@c001]$ ml R/3.6.1

[username@c001]$ Rscript program_name.r

 "Hello World!!"
```

If you are currently on a compute node and would like to switch back to the login node then please enter the `exit` command as follows:

```
[username@c001]$ exit
```

Running the script in Batch-Mode**: A sample Slurm batch job-script to run the serial program_name.r in batch mode is shown in Listing 2. This script should be run from a login node.**

```
#!/bin/bash
#SBATCH -J program_name
#SBATCH -o program_name.txt
#SBATCH -p compute1
#SBATCH -t 00:05:00
#SBATCH -N 1
#SBATCH -n 1

ml gnu8/8.3.0
ml R/3.6.1
Rscript program_name.r
```

Listing 2: Batch Job Script for R code (job_script16.slurm)

The job-script shown in Listing 2 can be submitted as follows:

```
[username@login001]$ sbatch job_script16.slurm
```

The output from the Slurm batch-job shown in Listing 2 can be checked by opening the output file as follows:

```
[username@login001]$ cat program_name.txt

 "Hello World!!"
```