



PARAM Ananta

Quick Start Guide

Ver. 1.2

Last updated: November 07, 2025

www.cdac.in

Copyright Notice

Copyright © 2021 Centre for Development of Advanced Computing
All Rights Reserved.

Any technical documentation that is made available by C-DAC (Centre for Development of Advanced Computing) is the copyrighted work of C-DAC and is owned by C-DAC. This technical documentation is being delivered to you as is, and C-DAC makes no warranty as to its accuracy or use. Any use of the technical documentation or the information contained therein is at the risk of the user. C-DAC reserves the right to make changes without prior notice.

No part of this publication may be copied without the express written permission of C-DAC.

Trademarks

CDAC, CDAC logo, NSM logo are trademarks or registered trademarks.

Other brands and product names mentioned in this manual may be trademarks or registered trademarks of their respective companies and are hereby acknowledged.



Intended Audience

This document is meant for PARAM Ananta users.

Typographic Conventions

Symbol	Meaning
<u>Blue underlined text</u>	A hyperlink or link you can click to go to a related section in this document or to a URL in your web browser.
Bold	The names of menus, menu items, headings, and buttons.
<i>Italics</i>	Variables or placeholders or special terms in the document.
<code>Console text</code>	Console commands



Getting help

For technical assistance or license renewal, please send an email to anantasupport@iitgn.ac.in

Give us your feedback

We value your feedback. Kindly send your comments on content of this document to anantasupport@iitgn.ac.in. Please include the page number of the document along with your feedback.



DISCLAIMER

The information contained in this document is subject to change without notice. C-DAC shall not be liable for errors contained herein or for incidental or consequential damages in connection with the performance or use of this manual.



Facility Location

PARAM Ananta Supercomputing Facility
Academic Block 5, Room 108-109
Indian Institute of Technology Gandhinagar
Palaj, Gujarat (382055), INDIA
Support Mail: anantasupport@iitgn.ac.in

Remote Access

Using SSH in Windows

To access PARAM Ananta, you need to “ssh” the login server. PuTTY is the most popular open source “ssh” client application for Windows, you can download it from (<http://www.putty.org/>). Once installed, find the PuTTY application shortcut in your Start Menu, desktop. On clicking the PuTTY icon, the PuTTY configuration dialog should appear. Locate the “Host Name or IP Address” input field in the PuTTY configuration screen. Enter the user’s name along with IP address or Hostname with which you wish to connect.

(e.g. [username]@paramananta.iitgn.ac.in -p 4422) for outside the campus

(e.g. [username]@paramananta.iitgn.ac.in – if accessing from within the campus

(e.g. [username]@10.0.62.167 -p 4422) for local access

Using SSH in Mac or Linux

Both Mac and Linux systems provide a built-in SSH client, so there is no need to install any additional package. Open the terminal, connect to SSH server by typing the following command:

```
ssh [username]@[hostname]
```

For example, to connect to the PARAM Ananta Login Node, with the username

```
user1: ssh user1@paramananta.iitgn.ac.in -p 4422
```

When logging into the PARAM Ananta server, additional security steps are in place to ensure safe access. Follow the process below carefully.

Step 1: Enter CAPTCHA

- Upon initiating your login, you will first be asked to **solve a CAPTCHA challenge**.
- This is a security measure to verify that the login attempt is being made by a human user.
- Carefully enter the characters displayed in the CAPTCHA prompt and press **Enter**.

Step 2: Enter Password

- After completing the CAPTCHA, you will be prompted to **enter your account password**.
- Type your password and press **Enter**.

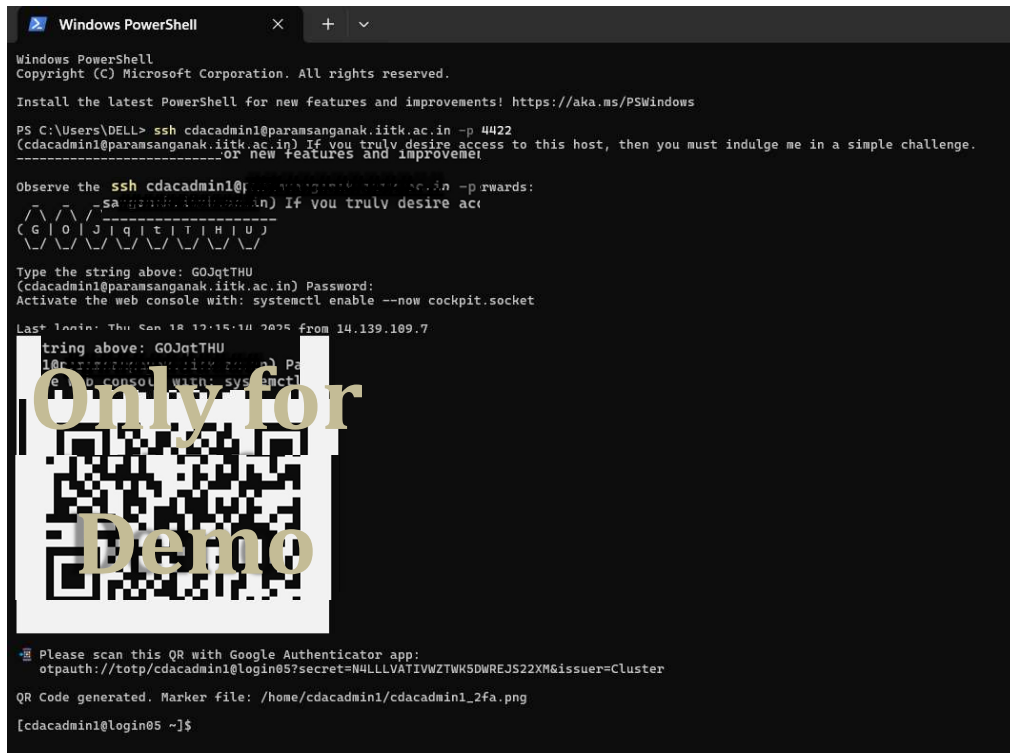
Step 3: Two-Factor Authentication (2FA) Setup

After entering your password, a **QR code** will appear in your terminal window.

This step is part of the **Two-Factor Authentication (2FA)** setup to enhance account security.

You will need to authenticate using an “Authenticator” app on your smartphone. **Google Authenticator** is the most popular open source “Authenticator” application for your smartphone, you can download it from.

- **Android (Google Play Store)** → [Google Authenticator](#)
- **iOS (Apple App Store)** → [Google Authenticator](#)



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\DELL> ssh cdacadmin1@paramsanganak.iitk.ac.in -p 4422
(cdacadmin1@paramsanganak.iitk.ac.in) If you truly desire access to this host, then you must indulge me in a simple challenge.
-----Or new features and improvements-----

Observe the ssh cdacadmin1@paramsanganak.iitk.ac.in -p 4422
(cdacadmin1@paramsanganak.iitk.ac.in) If you truly desire access to this host, then you must indulge me in a simple challenge.
-----Or new features and improvements-----

Type the string above: GOJgtTHU
(cdacadmin1@paramsanganak.iitk.ac.in) Password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Thu Jan 18 17:15:14 2024 from 14.139.189.7

Please scan this QR with Google Authenticator app:
otpauth://totp/cdacadmin1@login05?secret=N4LLLVAT1VMZTWK5DWREJS22XM6&issuer=Cluster
QR Code generated. Marker file: /home/cdacadmin1/cdacadmin1_2fa.png
[cdacadmin1@login05 ~]$
```

Figure 1 - A snapshot of the PARAM Ananta Login Page

Step 4: Link PARAM Ananta to Your Authenticator App

1. Open the Google Authenticator app on your smartphone → tap the “+” button → select Scan a QR code.
2. Use your phone’s camera to scan the QR code displayed in your terminal. Once scanned, a new entry will be created in the app — for example:

(e.g. Cluster: admin1@login)
3. Google Authenticator will generate a **6-digit one-time code** that **changes every 30 seconds**.

If the code is correct, your authentication will be successful, and you will be securely connected to the **PARAM Ananta** server.

Password

How to change the user password?

Use the **passwd** command to change the password (i. e. at least 13 characters) for the user from login node.

```
[test.user@login04 ~]$ passwd
#####
Password Requirements:
Minimum length: 13 characters
At least 1 uppercase letter
At least 1 lowercase letter
At least 1 number
At least 1 special character (e.g., !@#$%^&*)
#####

Changing password for user test.user.
(current) LDAP Password:
```

Figure 2 - A snapshot of changing user password

Regenerating QR Code or Secret Key (Google Authenticator Reset)

If you ever lose access to your **Google Authenticator app** (for example, due to a lost phone or a reinstall), you can reset and regenerate your authentication setup on the cluster.

Step 1: Run the Google Authenticator Command

Log in to the cluster and run the following command:

```
[username@login02 ~]$ google-authenticator
```

This will:

- Generate a new **secret key**
- Display a **new QR code** (scan this with Google Authenticator app)
- Provide backup codes (**emergency codes**)

Step 2: Follow the Prompts

During the setup process, you'll be asked a few configuration **questions**. Respond to them as shown below for the recommended secure setup:

- Update `.google_authenticator` file → **y**
- Disallow multiple uses → **y**
- Increase window size → **y**
- Enable rate-limiting → **y**

A file `~/.google_authenticator` will be created in your home directory.

⚠ **Important:** Save the 4–5 emergency backup codes (**emergency codes**) somewhere safe (e.g., offline notebook or password manager). These can be used if your phone is unavailable.

Transferring files between local machine and HPC cluster

Users need to have the data and application related to their project/research work on PARAM Ananta.

To store the data special directories have been made available to the users with the name “home” “scratch” the path to this directory is “/home” and “/scratch”. Whereas these directories are common to all the users, a user will get his own directory with their username in /home/, /scratch directories where they can store their data.

/home/<username>/: This directory is generally used by the user to install applications.
/scratch/<username>/: This directory is generally used by the user to schedule the jobs.

However, there is a limit to the storage provided to the users, the limits have been defined according to quota over these directories, all users will be allotted the same quota by default.

- /home: 50 GB
- /scratch: 200GB

When a user wishes to transfer data from their local system (laptop/desktop) to the HPC system, they can use various methods and tools.

A user using ‘Windows’ operating system will get methods and tools that are native to Microsoft Windows and tools that could be installed on your Microsoft windows machine. Linux operating system users do not require any tool. They can just use the “scp” command on their terminal, as mentioned below.

Users are advised to keep a copy of their data with themselves, once the project/research work is completed by transferring the data in from PARAM Ananta to their local system (laptop/desktop).

Step1: Downloading Data from PARAM Ananta (PARAM → Local/Remote System)

You can copy files from PARAM Ananta to either your Windows local machine or any remote Linux system.

a) From PARAM Ananta to Windows Local Machine: Run this command on your local Windows machine (PowerShell, Git Bash, or WSL):

Example:

```
scp -P 4422 -r username>@paramananta.iitgn.ac.in:<path to directory on HPC where to save the data> <path to the local data directory(e.g. /drives/c/Users/username/Downloads/)>
```

b) From PARAM Ananta to a Remote Linux Machine: Run this command **on your remote Linux machine**:

Example:

```
scp -P 4422 -r username>@paramananta.iitgn.ac.in:<path to directory on HPC where to save the data> <path to the local data directory(e.g. /root/)>
```

Step2: Uploading Data to PARAM Ananta (Local/Remote System → PARAM)

You can upload files to PARAM Ananta either from your Windows local machine or from any remote Linux system.

a) From Windows Local Machine to PARAM Ananta: Run this command on your local Windows machine (PowerShell, Git Bash, or WSL):

Example:

```
scp -P 4422 -r <path to the local data directory> username>@paramananta.iitgn.ac.in:<path to directory on HPC where to save the data>
```

b) From Remote Linux Machine to PARAM Ananta: Run this command **on your remote Linux machine**:

Example:

```
scp -P 4422 -r <path to the local data directory> username>@paramananta.iitgn.ac.in:<path to directory on HPC where to save the data>
```

Note: The Local system (laptop/desktop) should be connected to the network with which it can access the HPC system.

Tools

MobaXterm (Windows installable application):

It is a third party freely available tool which can be used to access the HPC system and transfer file to PARAM Ananta system through your local systems (laptop/desktop).

Link to download this tool: <https://mobaxterm.mobatek.net/download-home-edition.html>

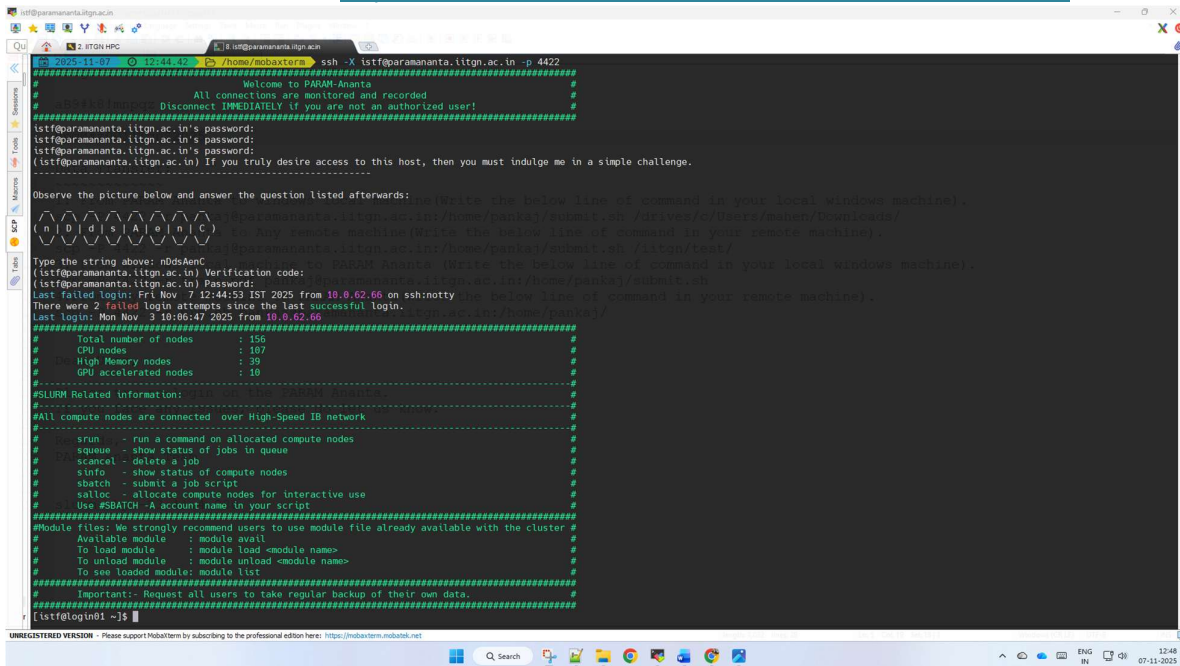


Figure 3 - A snapshot of command using MobaXterm

WinSCP (Windows installable application)

This popular tool is freely available and is used very often to transfer data from Windows machine to Linux machine. This tool is GUI based which makes it very user-friendly.

Link for this tool is: <https://winscp.net/eng/download.php>

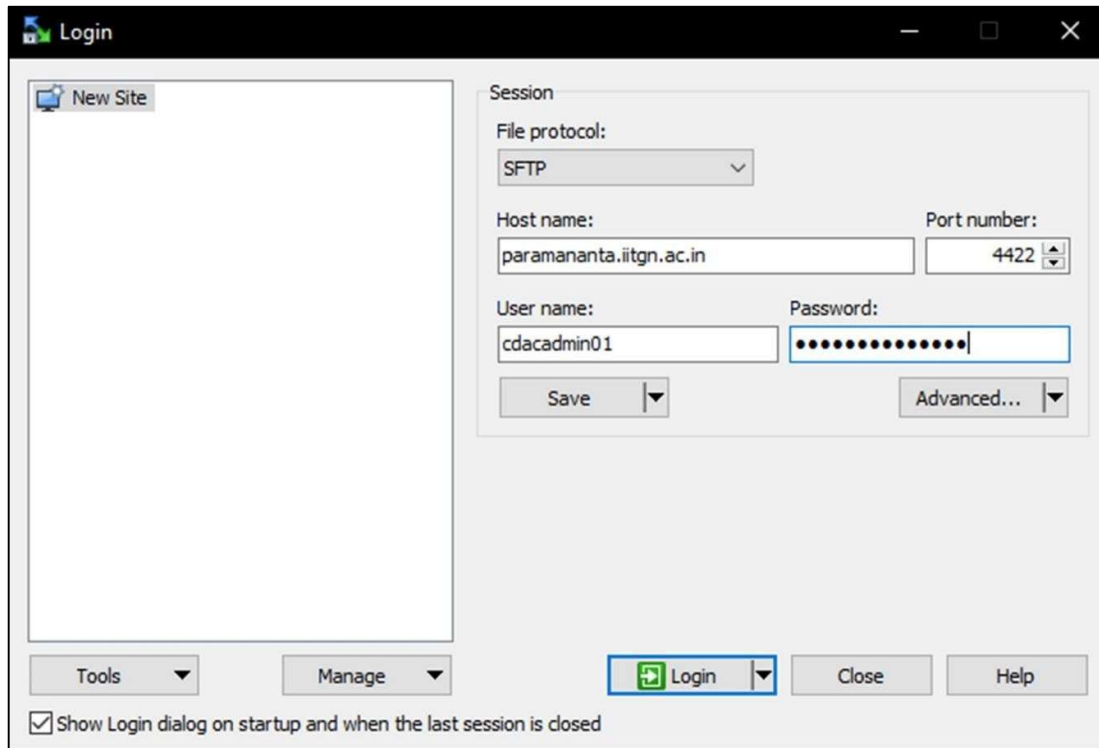


Figure 4 - A snapshot of "scp" tool to transfer file to and from remote computer.

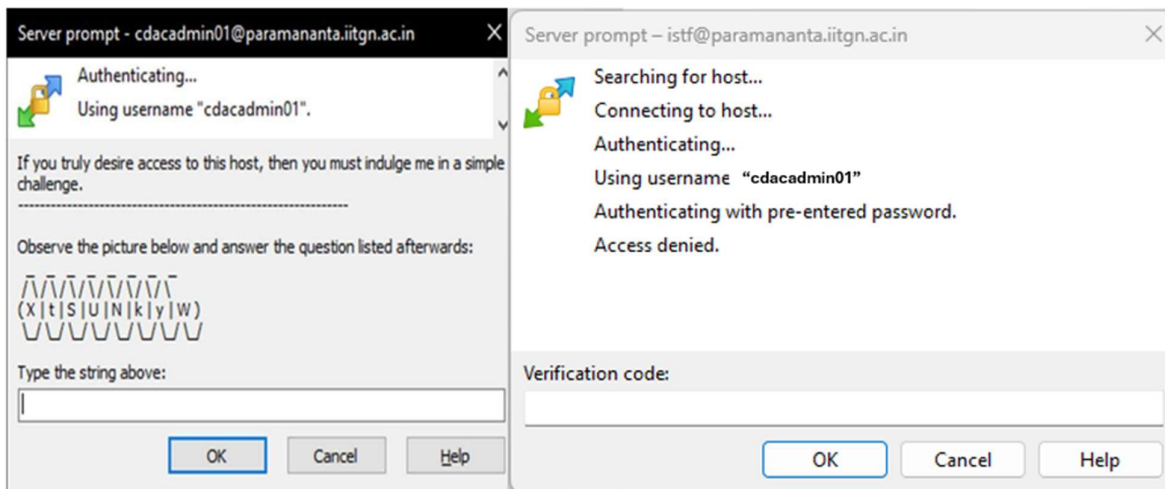


Figure 5— Enter Captcha/String

Figure 6— Enter Verification code(2FA code)

Note: Port Used for SFTP connection is 4422 and not 22. Please change it to 4422

Running Interactive Jobs

In general, the jobs can be run in an interactive manner or in batch mode. You can run an interactive job as follows:

The following command asks for a single core on one hour with default amount of memory.

```
$ srun --nodes=1 --ntasks-per-node=1 --time=01:00:00 --pty bash -i
```

The command prompt will appear as soon as the job starts. This is how it looks once the interactive job starts:

```
srun: job xxxxx queued and waiting for resources srun: job xxxxx has  
been allocated resources
```

Where xxxxx is the job id.

Exit the bash shell to end the job. If you exceed the time or memory limits the job will also abort.

Please note that PARAM Ananta is NOT meant for executing interactive jobs. However, for the purpose of quickly ascertaining successful run of a job before submitting a large job in batch (with large iteration counts), this can be used. This can even be used for running small jobs. The point to be kept in mind is that, since others too would be using this node, it is prudent not to inconvenience them by running large jobs.

It is a good idea to specify the CPU account name as well (if you face any problems)

```
$ srun --account=<NAME_OF_MY_ACCOUNT> --nodes=1 --ntasks-per-node=1 --  
time=01:00:00 -- pty bash -i
```

Managing Jobs through its Lifecycle

PARAM Ananta extensively uses modules. The purpose of module is to provide the production environment for a given application, outside of the application itself. This also specifies which version of the application is available for a given session. All applications and libraries are made available through module files. A User has to load the appropriate module from the available modules. User can add particular module in their ~/.bashrc also if they don't want to load particular module file for each time after they login.

module avail	# This command lists all the available modules
module load compiler/intel/2018.2.199	# This will load the intel compilers into your environment
module unload compiler/intel/2018.2.199	# This will remove all environment setting related to intel-2018 compiler loaded previously

A simple Slurm job script

```
#!/bin/sh
#SBATCH -N 1                                // specifies number of nodes
#SBATCH --ntasks-per-node=48                // specifies cores per node
#SBATCH --time=06:50:20 // specifies maximum duration of run
#SBATCH --job-name=lammps                   // specifies job name
#SBATCH --error=job.%J.err_node_48         // specifies error file name
#SBATCH --output=job.%J.out_node_48        //specifies output file name
#SBATCH --partition=standard                // specifies queue name

cd $SLURM_SUBMIT_DIR // To run job in the directory from where it is
submitted
export I_MPI_FABRICS=shm:dapl               //For Intel MPI versions 2019
onwards this value must be shm:ofi
mpiexec.hydra -n $SLURM_NTASKS lammps.exe
```

walltime

Walltime parameter defines as to how long your job will run. The maximum runtime of a job allowed as per QoS policy. If more than 10 days are required, a special request needs to be sent to HPC coordinator and it will be dealt with on a case-to-case basis. The command line to specify walltime is given below.

```
srun -t walltime <days-hours:mins:seconds>
```

and also, as part of the submit scripts described in the manual. If a job does not get completed within the walltime specified in the script, it will get terminated.

The biggest advantage of specifying appropriate walltime is that the efficiency of scheduling improves resulting in improved throughput in all jobs including yours. You are encouraging to arrive at the appropriate walltime for your job by executing your jobs few times.

NOTE: You are requested to explicitly specify the walltime in your command lines and scripts.

PARAM Ananta Queuing System

The sinfo command displays information about available nodes and partitions (queues).

The below figure includes details such as the partition name, maximum walltime, nodelist, and the maximum CPU, GPU, storage, and number of jobs allowed per user.

PARAM ANANTA Queuing System			
Partition Name	Timelimit	Nodelist	Max CPU/GPU/Storage/Job per user
DEBUG	5 Hours	cn[001-002]	<ul style="list-style-type: none">• 24 CPU cores per user.• 240 CPU cores per user.• 2 GPU cards per user• No job limit has been enforced.• For storage, every user has: 200 GB on scratch and 50 GB on home
SMALL	3 Days	cn[003-020]	
MEDIUM	5 Days	cn[021-060]	
LARGE	10 Days	cn[061-110]	
HIGHMEMORY	10 Days	hm[001-039]	
GPU	2 Days	gpu[001-010]	

Figure 7- List Partition

Submit the job

We can consider four cases of submitting a job.

1. Submitting a simple standalone job

This is a simple submit script which is to be submitted

```
$ sbatch slurm-job.sh
Submitted batch job 106
```

2. Submit a job that's dependent on a prerequisite job being completed

Consider a requirement of pre-processing a job before proceeding to actual processing. Pre-processing is generally done on a single core. In this scenario, the actual processing script is dependent on the outcome of pre-processing script.

Here's a simple job script. Note that the "slurm -j" option is used to give the job a name.

```
#!/usr/bin/env bash
#SBATCH -p standard
#SBATCH -J simple
sleep 60
Submit the job: $ sbatch simple.sh
Submitted batch job 149
```

Now we'll submit another job that's dependent on the previous job. There are many ways to specify the dependency conditions, but the "singleton" method is the simplest. The slurm -d singleton argument tells slurm not to dispatch this job until all previous jobs with the same name have completed.

```
$ sbatch -d singleton simple.sh //may be used for first pre-processing on
a core and then submitting
Submitted batch job 150
$ squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
    150 standard   simple user1  PD  0:00   1 (Dependency)
    149 standard   simple user1   R  0:17   1 cn001
```

Once the prerequisite job finishes the dependent job is dispatched.

```
$ squeue
JOBID PARTITION NAME USER ST TIME  NODES NODELIST(REASON)
    150 standard   simple user1   R  0:31   1 cn001
```

3. Submit a job with a reservation allocated

Slurm has the ability to reserve resources for jobs being executed by select users and/or select bank accounts. A resource reservation identifies the resources in that reservation and a time period during which the reservation is available. The resources which can be reserved include cores, nodes.

Use the command given below to check the reservation name allocated to your user account

```
$ scontrol show reservation
```

If your 'user account' is associated with any reservation the above command will show you the same. For e.g., the reservation name given is user_11. Use the command given below to make use of this reservation

```
$ sbatch --reservation=user_11 simple.sh
```

4. Submitting multiple jobs with minor or no changes (array jobs)

A **SLURM job array** is a collection of jobs that differs from each other by only a single index parameter. Job arrays can be used to submit and manage a large number of jobs with similar settings.

```
# Submit a job array with index values between 0 and 31
$ sbatch --array=0-31 -N1 tmp

# Submit a job array with index values of 1, 3, 5 and 7
$ sbatch --array=1,3,5,7 -N1 tmp

# Submit a job array with index values between 1 and 7
# with a step size of 2 (i.e. 1, 3, 5 and 7)
$ sbatch --array=1-7:2 -N1 tmp
```

Figure 8– Snapshot depicting the usage of “Job Array”

N1 is specifying number of nodes you want use for your job. example: N1 -one node, N4 - four nodes. Instead of tmp here you can use below example script.

```

#!/bin/bash
#SBATCH -N 1
#SBATCH --ntasks-per-node=48
#SBATCH --error=job.%A_%a.err
#SBATCH --output=job.%A_%a.out
#SBATCH --time=01:00:00
#SBATCH --partition=standard

module load compiler/intel/2018.2.199

cd /home/guest/Rajneesh/Rajneesh

export OMP_NUM_THREADS=${SLURM_ARRAY_TASK_ID}

/home/guest/Rajneesh/Rajneesh/md_omp

```

List jobs

Monitoring jobs on SLURM can be done using the command **squeue**. Squeue is used to view job and job step information for jobs managed by SLURM.

```

$ squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
  106 standard      slurm-job user1  R   0:04      1 cn001

```

Get job details

scontrol can be used to report more detailed information about nodes, partitions, jobs, job steps, and configuration.

scontrol show node - shows detailed information about compute nodes.

```

[root@login04 ~]# scontrol show node cn001
NodeName=cn001 Arch=x86_64 CoresPerSocket=24
  CPUAlloc=0 CPUTot=48 CPULoad=0.01
  AvailableFeatures=cpu,centos7
  ActiveFeatures=cpu,centos7
  Gres=(null)
  NodeAddr=cn001 NodeHostName=cn001 Version=20.11.8
  OS=Linux 3.10.0-1160.el7.x86_64 #1 SMP Mon Oct 19 16:18:59 UTC 2020
  RealMemory=1 AllocMem=0 FreeMem=182850 Sockets=2 Boards=1
  State=IDLE ThreadsPerCore=1 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
  Partitions=standard,cpu
  BootTime=2021-12-15T12:42:46 SlurmdStartTime=2021-12-29T15:17:40
  CfgTRES=cpu=48,mem=1M,billing=48
  AllocTRES=
  CapWatts=n/a
  CurrentWatts=0 AveWatts=0
  ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
  Comment=(null)

```

Figure 9– scontrol show node displays compute node information

scontrol show partition - shows detailed information about a specific partition

```
[root@login04 ~]# scontrol show partition hm
PartitionName=hm
  AllowGroups=ALL AllowAccounts=ALL AllowQos=ALL
  AllocNodes=ALL Default=NO QoS=N/A
  DefaultTime=NONE DisableRootJobs=NO ExclusiveUser=NO GraceTime=0 Hidden=NO
  MaxNodes=UNLIMITED MaxTime=3-00:00:00 MinNodes=0 LLN=NO MaxCPUsPerNode=UNLIMITED
  Nodes=hm[001-039]
  PriorityJobFactor=75 PriorityTier=75 RootOnly=NO ReqResv=NO OverSubscribe=NO
  OverTimeLimit=NONE PreemptMode=OFF
  State=UP TotalCPUs=1872 TotalNodes=39 SelectTypeParameters=NONE
  JobDefaults=(null)
  DefMemPerNode=UNLIMITED MaxMemPerNode=UNLIMITED
```

Figure 10– scontrol show partition displays specific partition details

scontrol show job - shows detailed information about a specific job or all jobs if no job id is given.

```
[root@ananta0 ~]# scontrol show job 4253
JobId=4253 JobName=copy.sh
  UserId=atos01(40004) GroupId=atos01(40004) MCS_label=N/A
  Priority=19212 Nice=0 Account=atos QOS=normal
  JobState=RUNNING Reason=None Dependency=(null)
  Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
  RunTime=00:16:44 TimeLimit=02:00:00 TimeMin=N/A
  SubmitTime=2022-03-23T19:31:40 EligibleTime=2022-03-23T19:31:40
  AccrueTime=2022-03-23T19:31:40
  StartTime=2022-03-23T19:31:40 EndTime=2022-03-23T21:31:40 Deadline=N/A
  SuspendTime=None SecsPreSuspend=0 LastSchedEval=2022-03-23T19:31:40
  Partition=standard AllocNode:Sid=login03:91479
  ReqNodeList=gpu001 ExcNodeList=(null)
  NodeList=gpu001
  BatchHost=gpu001
  NumNodes=1 NumCPUs=1 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
  TRES=cpu=1,node=1,billing=1
  Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
  MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
  Features=(null) DelayBoot=00:00:00
  OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
  Command=/mnt/scratch/atos01/test/copy.sh
  WorkDir=/mnt/scratch/atos01/test
  StdErr=/mnt/scratch/atos01/test/slurm-4253.out
  StdIn=/dev/null
  StdOut=/mnt/scratch/atos01/test/slurm-4253.out
  Power=
  NtasksPerTRES:0
```

Figure 11– scontrol show job displays specific job information

scontrol update job - changes attributes of submitted job; like time limit, priority (root only)

```
$ scontrol show job 106
JobId=106 Name=slurm-job.sh
  UserId=user1(1001) GroupId=user1(1001)
  Priority=4294901717 Account=(null) QOS=normal
  JobState=RUNNING Reason=None Dependency=(null)
  Requeue=1 Restarts=0 BatchFlag=1 ExitCode=0:0
  RunTime=00:00:07 TimeLimit=14-00:00:0 TimeMin=N/A
  SubmitTime=2021-01-26T12:55:02 EligibleTime=2021-01-26T12:55:02
  StartTime=2021-01-26T12:55:02 EndTime=Unknown
  PreemptTime=None SuspendTime=None SecsPreSuspend=0
  Partition=standard AllocNode:Sid=atom-head1:3526
  ReqNodeList=(null) ExcNodeList=(null)
  NodeList=cn001
  BatchHost=cn001
  NumNodes=1 NumCPUs=2 CPUs/Task=1 ReqS:C:T=*:*:~
  MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
  Features=(null) Gres=(null) Reservation=(null)
  Shared=0 Contiguous=0 Licenses=(null) Network=(null)
  Command=/home/user1/slurm/local/slurm-job.sh
  WorkDir=/home/user1/slurm/local
```

scontrol update job= 106 TimeLimit=04-00:00:0

Suspend a job (root only):

```
# scontrol suspend 135
# squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
  135   standard simple.sh user01 S   0:10    1   cn001
```

Resume a job (root only):

```
# scontrol resume 135
# squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
  135   standard simple.sh user01 R   0:13    1   cn001
```

Kill a job. Users can kill their own jobs; root can kill any job.

```
$ scancel 135
$ squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
```

Hold a job:

```
$ squeue
JOBID PARTITION      NAME      USER  ST          TIME  NODES NODELIST(REASON)
   139      standard    simple  user01 PD           0:00      1  (Dependency)
   138      standard    simple  user01  R           0:16      1  cn001
$ scontrol hold 139
$ squeue
JOBID PARTITION      NAME      USER  ST          TIME  NODES NODELIST(REASON)
   139      standard    simple  user01 PD           0:00      1  (JobHeldUser)
   138      standard    simple  user01  R           0:32      1  cn001
```

Release a job:

```
$ scontrol release 139
$ squeue
JOBID PARTITION      NAME      USER  ST          TIME  NODES NODELIST(REASON)
   139      standard    simple  user01 PD           0:00      1  (Dependency)
   138      standard    simple  user01  R           0:46      1  cn001
```