**Assignment 13 – Project completion**

**TEAM BLACK**

**Team Members:**

* Akshata Nanjappa
* Nandikonda Srikanth
* Mayur Kenkre
* Rushikesh Deore

**GOAL:**

1. Complete all the milestones to finish our project to establish Slurm configuration between the team server and our individual instances. Run complex jobs, in array and optimization.
2. Run array jobs on our individual instances from control node to display the content of files
3. Learning and contributions

**MILESTONE COMPLETION:**

**Milestone 1,2,3:** The 3 Milestones up to setting up the Slurm configuration has been completed in our previous class. We have successfully established our individual instances as the worked nodes and the team instance as the control node:

Diagram

Description automatically generated

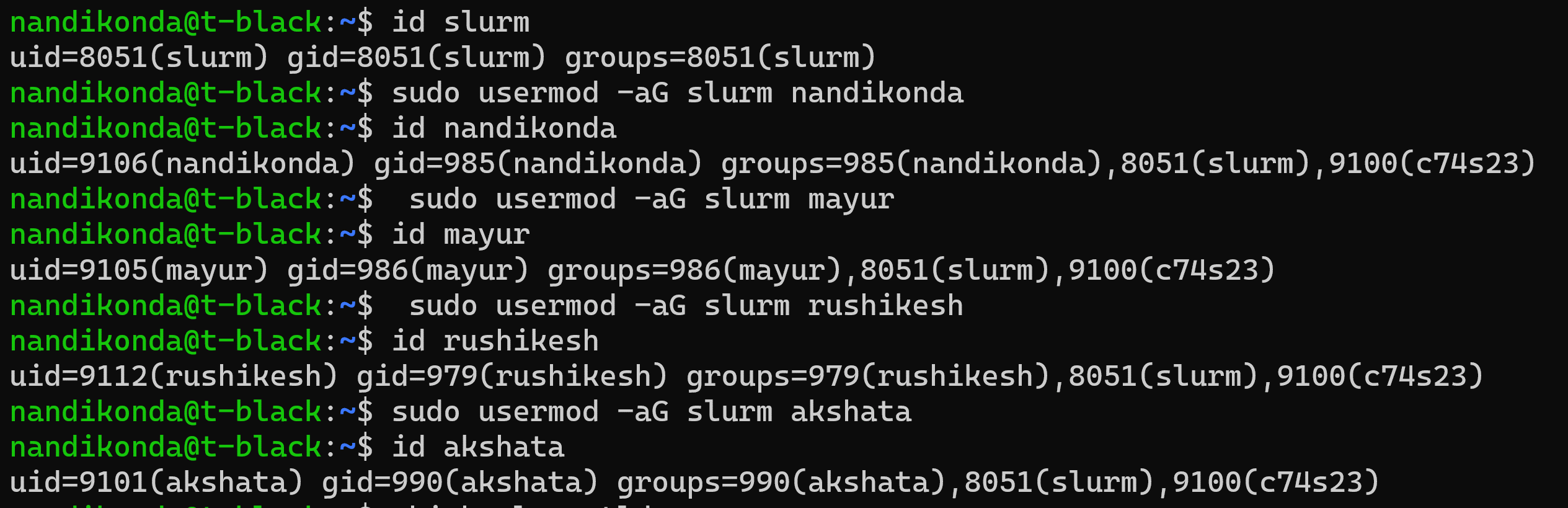
Text

Description automatically generated

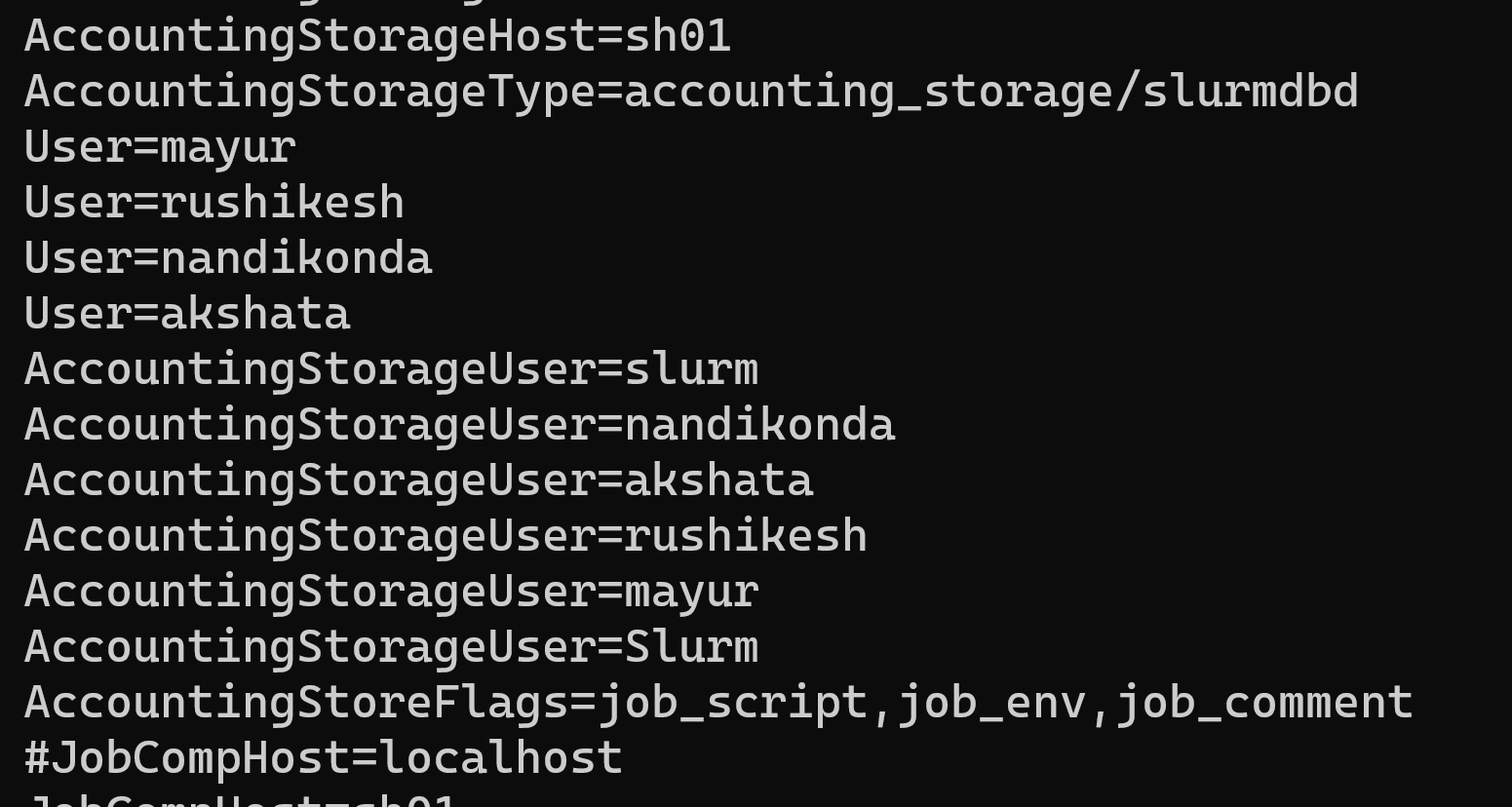
**MILESTONE 4: Account Provision**

Provision accounts on slurm:

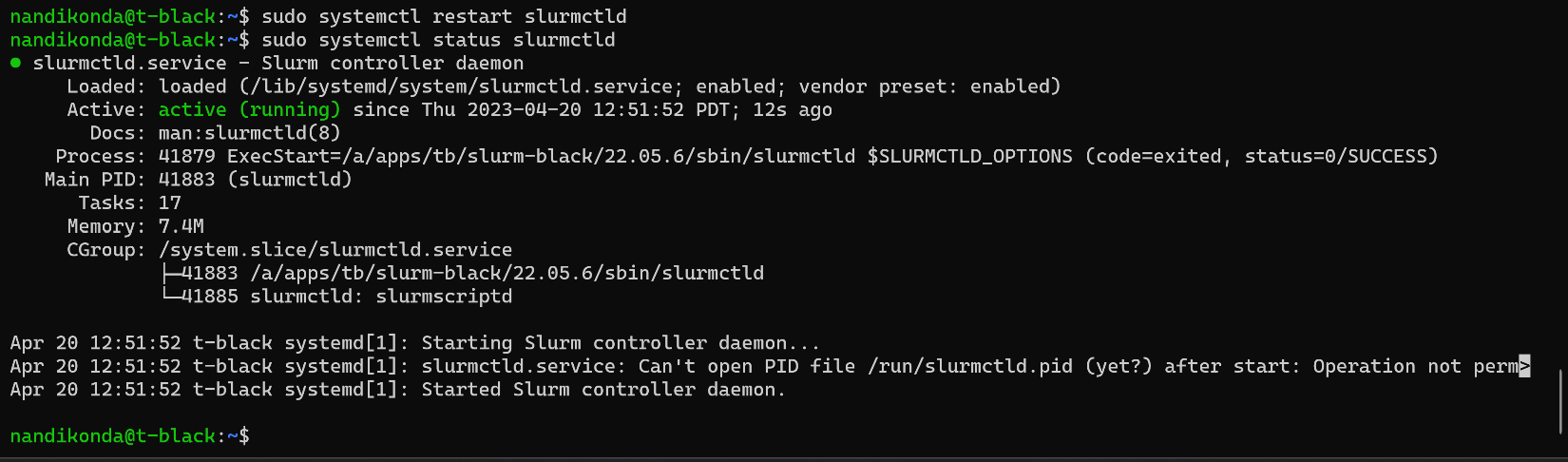
* Grant the necessary permissions to each user account of the team by adding them to the appropriate groups. I have added users to the "slurm" group so they can submit jobs to Slurm.



* Edited the Slurm configuration files to specify the user accounts and their associated permissions. The configuration files for the slurm for team black are located in the "/a/apps/tb/slurm-black/22.05.6/etc/slurm.conf" directory.



* Restart the Slurm controller service.



**MILESTONE 5:**

**Task 1: Integrate the Slurm controller and worker nodes**

Slurm controller and worker nodes were integrated in the milestone 2 and 3, we can verify if worker nodes are registered with the slurm controller by checking the state of slurm.

Text

Description automatically generated

From the output, we can confirm that the black cluster has two partitions named "ibk0" and "batch\*", with the same number of nodes available in each partition.

**Task 2: Test the integrated system and ensure it is functioning properly**

1. Submit a test job to the Slurm cluster using the "sbatch" command. We have created a simple script file named " firstjo.sh" with the following contents:

Text

Description automatically generated

1. Submit the job to the Slurm cluster:



1. Once the job has completed, check the output file (named "slurm-234.out" by default) to verify that it ran successfully.

Graphical user interface, text, application

Description automatically generated

**MILESTONE 6: Optimization, Scaling, and QOS**

Task 1: Set Quality of Service (QOS).

QOS stands for Quality of Service. QOS allows administrators to define various levels of service quality for different users or groups of users. The QOS feature in SLURM enables administrators to prioritize jobs based on their importance or resource requirements. We set the QOS as black and can see the same below. In the SLURM configuration file, setting QOS means defining the different quality of service levels that can be assigned to jobs submitted to the cluster ‘black’. We can set our own flags in the slurm.conf file basedon our cluster and job requirements.

1. Create a new QOS using the sacctmgr command

sacctmgr -i add qos black set MaxTRESPerUser=cpu=1 priority=10 Flags=DenyOnLimit MaxWall=0-0:5:0

This command will create a new qos maximum number of CPU cores that a user can request in a job as 1, priority as 10, Denies jobs if they exceed the limits set by the QoS and max wall time as 5 min.

Text

Description automatically generated

1. Have added an account to the Slurm accounting system using th below command

Text

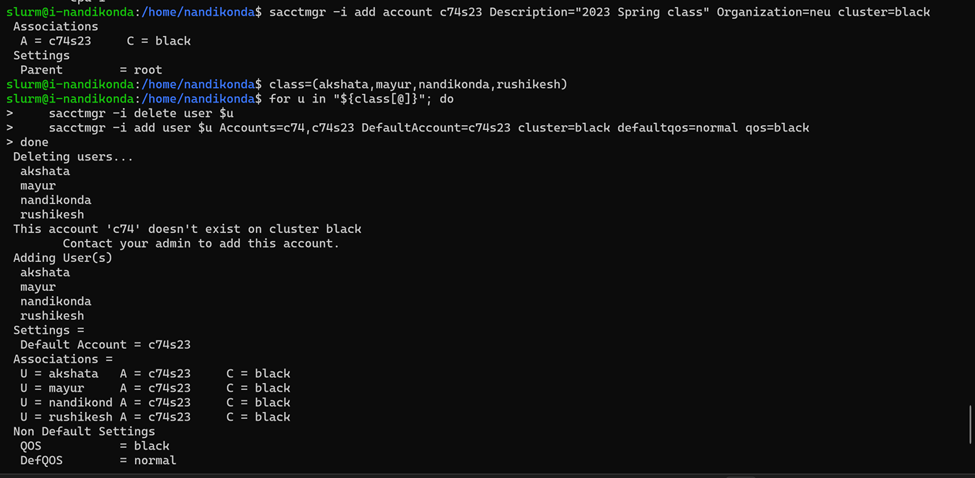
Description automatically generated

1. Verify that account is added

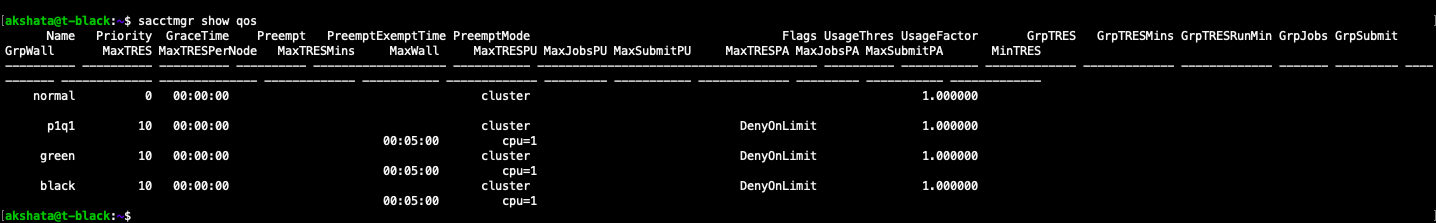
Text

Description automatically generated

1. Added our individual users to the c74 and c74s23 accounts and specified the cluster as black and qos as black. This will associated our user accounts to **black** qos



1. Verify if the qos is added by running the below command.



1. Verify the user association.

A picture containing text

Description automatically generated

**MILESTONE 7: ADVANCED PARALLEL JOBS**

Array jobs: Submitting array jobs via the control nodes helps decide how many arrays we run an array job on our worker nodes with the command below. Here, we use the job along with specifications like memory, nodes, time and so on with the below command for sub\_arr.sh bash script:

sbatch sub\_arr.sh

here, sinfo shows all our nodes being in the allocated state. Sbatch command ran job 205, and squeue shows the result of the jobs running

Text

Description automatically generated

Upon checking the output file, we can see:

The j\_205\_1, 2,3 and 4 have all 4 outputs from our 4 nodes and displays the result as needed from the script.

Text

Description automatically generated

Chain Submission:

Chain submission is a feature in the Slurm workload manager that allows multiple jobs to be submitted as a chain or a sequence, where the output of one job serves as the input for the next job in the chain.

We have sub1.sh and sub2.sh where upon successful completion of sub1.sh, sub2.sh runs and displays the output text file contents that is run in the first job. Here also we will be able to mention the memory, nodes, CPU and more as needed.

Below is the output upon running chain.sh to run the 2 jobs on a worker node:

Text

Description automatically generated with low confidence

Graphical user interface, text, application, chat or text message

Description automatically generated

**INDIVIDUAL ARRAY JOB SUBMISSION:**

Here, we have executed the script below where each array task outputs the matching file name. By using

--array flag with sbatch command we can run array jobs.

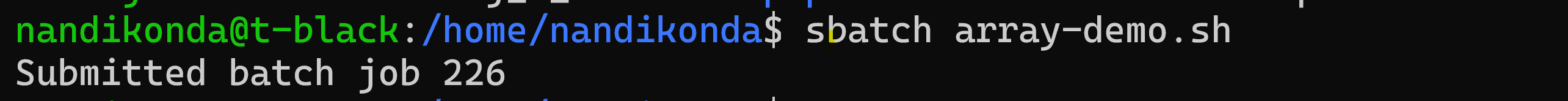
An array job submission is a feature of SLURM, that allows a user to submit multiple jobs with similar requirements as a single job array. With an array job submission, a user can define a range or list of task IDs, and the job scheduler will create and manage individual tasks or sub-jobs for each ID in the list or range.

We are running a simple bash script where the output is saved as the slurm array task ID and in the output the job ID is saved as 2 parts giving the outputs as the array or file numbers, respectively.

Text

Description automatically generated

Upon executing this, we can see the output as below:



The job runs very quickly, and we can see the generated output files:

Text

Description automatically generated

Above, we can see j\_226\_1 and j\_226\_2 files being created, and these contain the content as request in the output.

**LEARNING:**

The takeaway from the course was a major knowledge on how infrastructure works on an Ubuntu OS. We learnt several Devops concepts concentrating on fields of job scheduling, containerization, using Ansible and Slurm for running tasks on a cluster of servers.

Overall, with these milestones we were able to build a scalable infrastructure where we can run multiple jobs on worker nodes using a control node with slurm. This architecture can be orchestrated according to the needs of a team, we can run heavy jobs by altering the CPU, memory and more. We learned how both easy and complex tasks can be executed using this design and method.

**CONTRIBUTIONS:**

As a team we spent a considerable time sitting and understanding the basic requirements and process for the project. And when it comes to how we planned and split the tasks

* **Srikanth:** Slurm account configurations, part of the document, presentations
* **Akshata:** Setting up the array and chain jobs, Optimization, documentations
* **Mayur:** QOS set up, project wiki update, Debugging and Testing
* **Rushikesh:** Scaling the architecture, Integration testing.