User Manual for DGX

(1) How to Login on DGX proxy server.

Now users have to login on 192.168.121.190 (DGX Proxy Server) with their same credentials as they use to do on main DGX system.

For Example:

```
[2020-06-25 16:52.58]  ~
[manokattyagi.manokattyagi-PC] ➤ ssh admin_icc@192.168.121.190
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-142-generic x86_64)
```

And after login you will be in your shell.

```
Last login: Thu Jun 25 16:49:18 2020 from 10.61.92.9
admin_icc@dgx1-proxy:~$
```

After Login you need to go to your code location/directory or path as shown below.

Here I am in my home directory where my scripts are resides. And a Slurm Job Scheduler is running through which users has to submit their jobs.

```
admin_icc@dgx1-proxy:~$ pwd
/home/admin_icc
admin_icc@dgx1-proxy:~$
admin_icc@dgx1-proxy:~$
admin_icc@dgx1-proxy:~$ ls
adduser.sh  NVIDIA_CUDA-10.1_Samples  script.sh  ssh.tar  test.sh         test_tf_63.out
cmd         result                    simple.sh  testing  test_tf_62.out  test_tf_64.out
admin_icc@dgx1-proxy:~$
```

Slurm Overview

SLURM (Simple Linux Utility for Resource Management) is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters. SLURM requires no kernel modifications for its operation and is relatively self-contained. As a cluster workload manager, SLURM has three key functions.

First, it allocates exclusive and/or non-exclusive access to resources (compute nodes) to users for some duration of time so they can perform work.
Second, it provides a framework for starting, executing, and monitoring work (normally a parallel job) on the set of allocated nodes.
Finally, it arbitrates contention for resources by managing a queue of pending work.

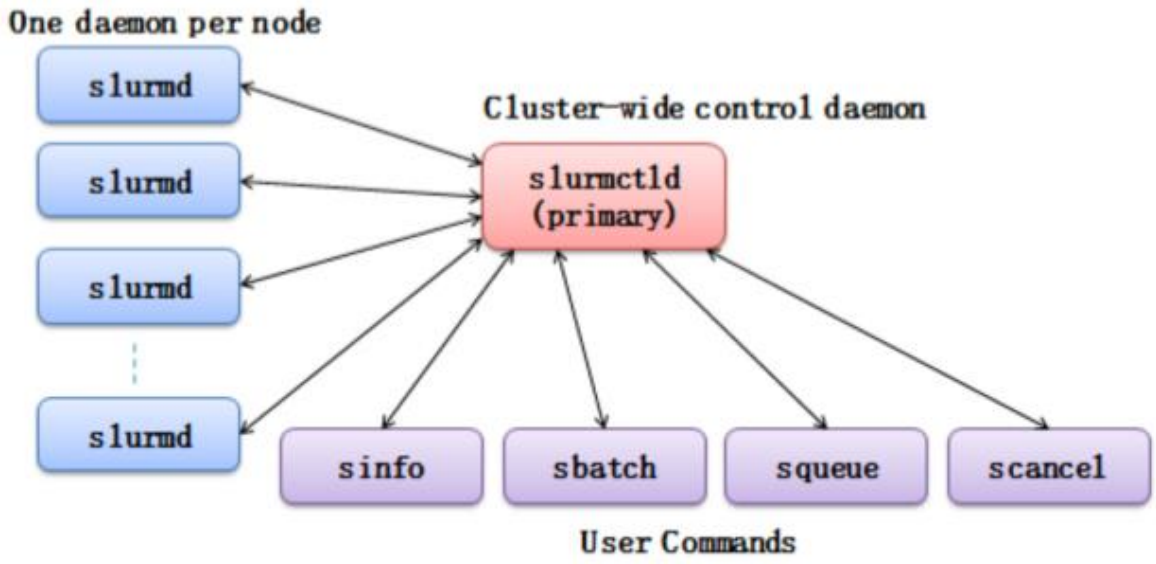


Figure: SLURM Architecture

SLURM Commands for the User.

1. SBATCH: Submit a batch script to SLURM
sbatch < SCRIPT NAME >

2. SQUEUE: View information about jobs located in the SLURM scheduling queue.
2.1 SQUEUE is used to view job and job step information for jobs managed by SLURM.
squeue < JOB ID >

3. SINFO: View information about SLURM nodes and partitions. SINFO is used to view
3. partition and node information for a system running SLURM.
sinfo

4. SCANCEL: Used to signal jobs or job steps that are under the control of SLURM
scancel < JOB ID >

Sample SLURM Job Script

#!/bin/sh #SBATCH --job-name=tf_job_test
Job name #SBATCH --ntasks=1
Run on a single CPU #SBATCH --time=02:00:00
Time limit hrs:min:sec
#SBATCH --output=tf_test_%j.out
Standard output and error log
#SBATCH --cpus-per-task=1
#SBATCH --gres=gpu:1
#SBATCH --mem=32GB
echo $CUDA_VISIBLE_DEVICES
NV_GPU=$CUDA_VISIBLE_DEVICES nvidia-docker run -t ${USE_TTY} --name
$SLURM_JOB_ID -user $(id -u):$(id -g) -rm -v /home/$USER:/home/$USER
nvcr.io/nvidia/tensorflow:19.04-py3 python -c 'import tensorflow as tf;
print(tf.__version__)'

Job Submission example through Slurm.

(1) Before Submitting Your job check the system environment. The system should be in idle state as shown below.

```
admin_icc@dgx1-proxy:~$
admin_icc@dgx1-proxy:~$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
dgx*         up   infinite      1   idle iitrdgx
admin_icc@dgx1-proxy:~$
```

(2) After that verify if there any jobs are running on system. As shown below.

```
admin_icc@dgx1-proxy:~$
admin_icc@dgx1-proxy:~$ squeue -a
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
admin_icc@dgx1-proxy:~$
```

(3) As shown above none of the job is running on the system. Now I have submitted a sample job as shown below.

(4) Important Note: Before submitting your job through slurm every user has to mention the memory value as suggested bellow (128GB). Whether your program/code need lesser than 128GB or greater than 128GB in both condition you cannot exceed the maximum value of 128GB. Otherwise the user's job will not run and will be queued.

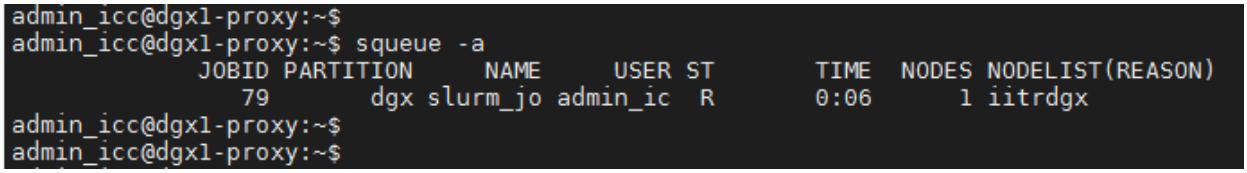
```
#SBATCH --mem=128GB
```

Job submitted higher than 128 GB would be queued as below.

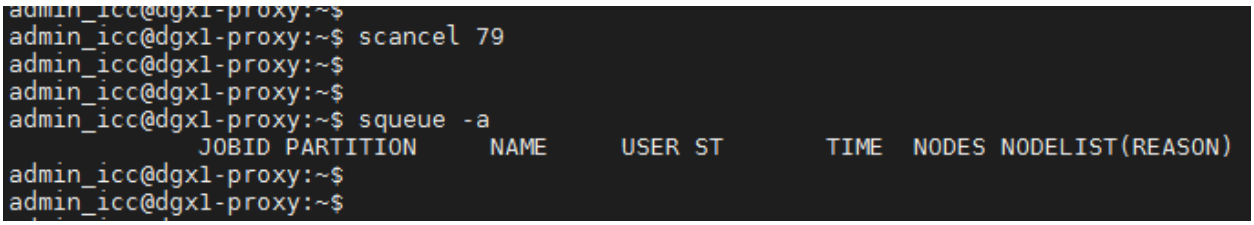
```
609      dgx slurm_jo   locuz PD       0:00      1 (QOSMaxMemoryPerJob)
```

```
admin_icc@dgx1-proxy:~$ ls
adduser.sh  NVIDIA_CUDA-10.1_Samples  script.sh  ssh.tar  test.sh         test_tf_63.out  test_tf_76.out
cmd         result                    simple.sh  testing  test_tf_62.out  test_tf_64.out  test_tf_78.out
admin_icc@dgx1-proxy:~$
admin_icc@dgx1-proxy:~$
admin_icc@dgx1-proxy:~$
admin_icc@dgx1-proxy:~$
admin_icc@dgx1-proxy:~$ sbatch  simple.sh
Submitted batch job 79
admin_icc@dgx1-proxy:~$
```

(5) After submitting the job when you will check the status of your job, it should show in running state as shown below.



(6) And if you want to stop/kill your Job then enter this command as shown below.



How to run docker and create own images.

(1.) User can create Dockerfile.
Example Dockerfile:
FROM ubuntu
RUN apt update && apt install -y cowsay
CMD ["/usr/games/cowsay", "Hello!"]

(2.) User docker build command on Slurm. User must be inside directory where Dockerfile available. Need to mention same on Slurm Script.
Example command:
cd /home/test/Dockerfile
docker build -t cowsay:v1 .

(3.) Use docker run command on Slurm.

Example Command:
nvidia-docker run -t ${USE_TTY} --name $SLURM_JOB_ID --rm cowsay:v1

(4.) Use docker run command on Slurm.
Example Command:
nvidia-docker run -t ${USE_TTY} --name $SLURM_JOB_ID --rm cowsay:v1

User can pass comand during docker run. For example,

nvidia-docker run -t ${USE_TTY} --name $SLURM_JOB_ID --user $(id -u $USER):$(id -g $USER) --rm -v /home/test:/workspace nvcr.io/nvidia/tensorflow:18.07-py3 python -c 'import tensorflow as tf; print(tf.__version__)'
python -c 'import tensorflow as tf; print(tf.__version__)'

Thank You