

# Volta Cluster User Guide

April 2020

By Ku Wee Kiat, HPC Specialist, NUS IT



**READ ALL SECTIONS OF THIS GUIDE  
CAREFULLY**

**MOST SUPPORT REQUESTS &  
QUESTIONS FROM USERS ARE  
ALREADY ANSWERED WITHIN THIS  
GUIDE**

- Access
- Resources
- Short Interactive Jobs
- Batch Jobs
- Multi-GPU Scaling
- Python Packages
  - Install Python Packages in User Package Path (Default)
  - Install Python Packages in Custom Path
- PBS Job Scheduler
- Tips and Tricks
- Debugging
- FAQ/Common Problems
- nTouch Support Request Template

# Access

# Access

- Login via ssh to NUS HPC login nodes
  - atlas9
  - atlas7-c10.nus.edu.sg
  - atlas8-c01.nus.edu.sg
- If you are connecting from **outside NUS network**, please connect to **Web VPN** first
  - <http://webvpn.nus.edu.sg>

# Access

OS	Access Method	Command
Linux	ssh from terminal	<code>ssh nusnet_id@atlas8-c01.nus.edu.sg</code>
MacOS	ssh from terminal	<code>ssh username@hostname</code>
Windows	ssh using mobaxterm or putty	<code>ssh username@hostname</code>

```
[2018-11-12 11:44.47] ~  
[ccekwk.6620G] > ssh ccekwk@atlas8.nus.edu.sg  
Warning: Permanently added 'atlas8.nus.edu.sg' (RSA) to the list of known hosts.  
ccekwk@atlas8.nus.edu.sg's password:  
Last login: Mon Nov 12 10:04:01 2018 from 172.23.191.235  
*****  
# Use PBS Job Scheduler to Submit and Manage Jobs    #  
#                                                     #  
# Help info available via command: hpc pbs -help     #  
*****  
[ccekwk@atlas8-c01 ~]$
```

# File Transfer

1. MobaXterm built-in sftp client
2. Filezilla client
3. Linux/Mac OS Terminal Tools
  - a. scp
  - b. rsync
  - c. sftp



# Resources

# Resources: Hardware

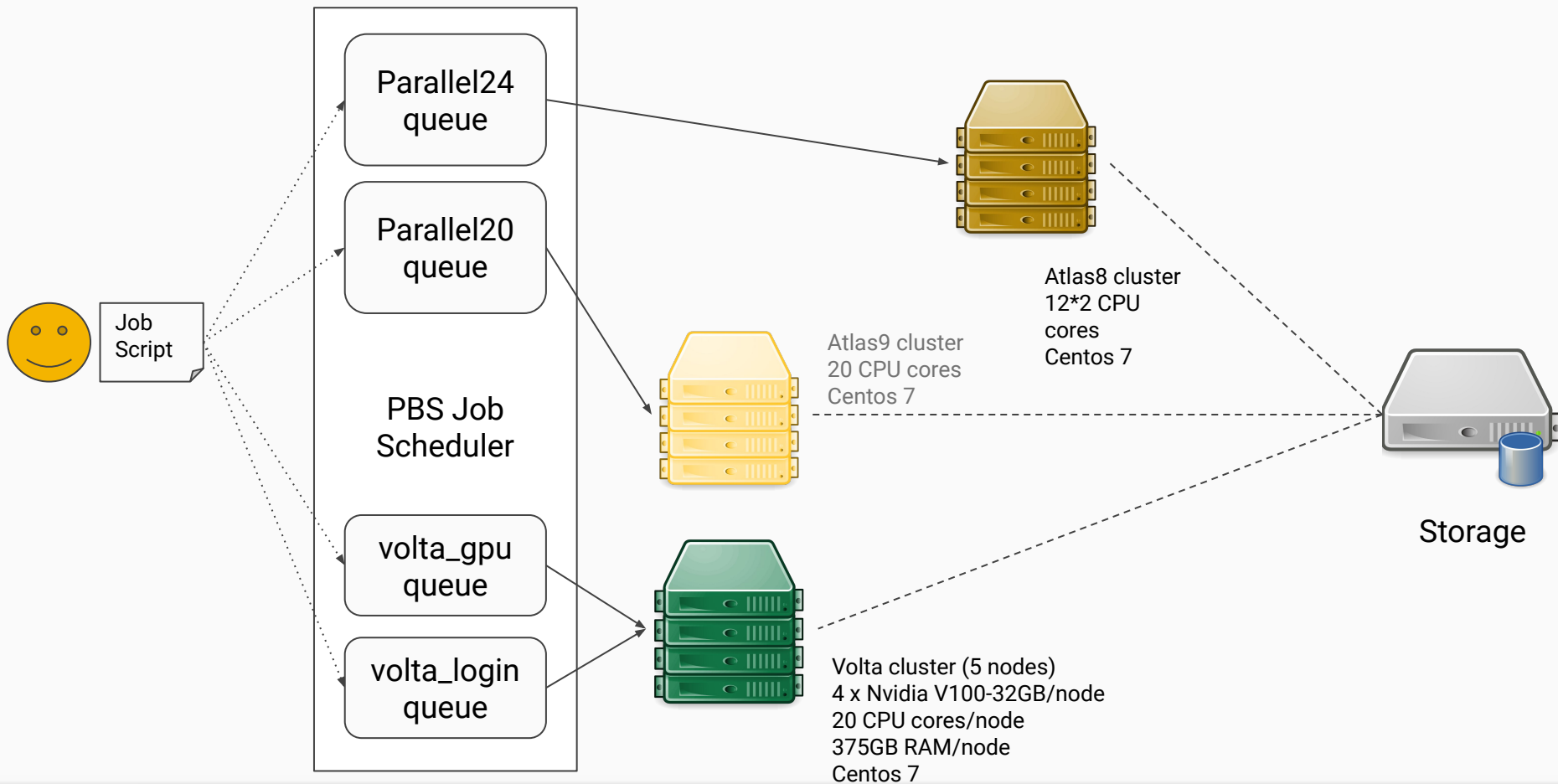
- Standard CPU HPC Clusters
  - Atlas 8
  - Atlas 9
- GPU Clusters
  - 5 nodes x 4 Nvidia Tesla V100-32GB

No internet access on Volta Servers

## Resources: Hardware/Storage

Directories	Feature	Disk Quota	Backup	Description
/home/svu/\$USERID	Global	20 GB	Snapshot	Home Directory. U:drive on your PC.
/hpctmp/\$USERID	Local on All Atlas/Volta cluster	500 GB	No	Working Directory. <b>Files older than 60 days are purged automatically</b>
/scratch/\$USERID	Local to each Volta node	5 TB	No	For quick read/write access to datasets. Create a folder with your NUSNET ID. <b>Routinely purged.</b>
/scratch2/\$USERID	Available on Atlas 9 and Volta Cluster		No	For quick read/write access to datasets. Create a folder with your NUSNET ID. <b>Routinely purged.</b>

Note: Type “hpc s” to check your disk quota for your home directory



# Resources: Containers

**IMPORTANT**

Each Deep Learning framework has its own [singularity](#) container:

```
l[ccekwk@atlas9-c01 ~]$ ls /app1/common/singularity-img/3.0.0/
caffe_0.17.2_nvcr_19.01-py27.simg  darknet_jimmylin.simg  lgbm_v2.simg  rapidsai_5.0_nvcr_c10_u16.04-py36.simg
caffe2_0.8.1_nvcr_18.08-py27.simg  intel_mkl_tensorflow_1.11.0_cpu_py27.simg  openfoamplus_v1806.sig  tensorflow_1.12_nvcr_19.01-py3.simg
caffe2_0.8.1_nvcr_18.08-py35.simg  intel_mkl_tensorflow_1.11.0_cpu_py35.simg  pytorch_1.0.0_nvcr_19.01_py3.simg  tensorflow_1.8.0_nvcr_18.07-py35.simg
```

Containers are located in: [/app1/common/singularity-img/3.0.0/](#)

Please take a look in the above folder to find the container with the Deep Learning Library you need

**All scripts must be executed within a container.**

- For Ubuntu containers with CUDA/Nvidia Drivers and Python see the following folder: /app1/common/singularity-img/3.0.0/CUDA

# Resources: Containers

1. Contact us only if you have any customised container requests.
2. If you build your own container, **contact us** so we can check if it has the required and compatible CUDA libraries the GPUs

# Resources: Containers (Singularity)

- **Simple:** with all components **packaged in a single file**, including the definitions, configurations, metadata, and security keys.
- **Fast:** with **native support** for GPU.
- Possible to **Build your own Singularity Container** and upload for use on NUS HPC
  - <https://www.sylabs.io/docs/>

# Building Your Own Container

If you build your own container, please

1. ensure that you install cuda drivers/libraries
2. you **do not** have to **install GPU drivers** (this is provided by host system)
3. verify that it **can use GPUs**
  - a. use interactive queue:
  - b. `singularity exec your_container.simg nvidia-smi`

**IMPORTANT TO READ**



# Running a Container (Volta Nodes)

```
$ singularity exec $image bash
```

Substitute **\$image** with full path to container image file

E.g.:

```
$ singularity exec /app1/common/singularity-img/3.0.0/tensorflow_1.12_nvcr_19.01-py3.simg bash
```

Put the commands in a **job script**. It is required.

**This slide applies only to VOLTA nodes.**

**DO NOT RUN THE ABOVE IN ATLAS NODES**

# Types of Jobs

## 1. Short, Interactive Jobs

- a. For debugging
- b. For checking if your code works

## 2. Batch Jobs

- a. For running working code for long periods of time.

# Short or Interactive Jobs

# Short Interactive Jobs

Run short interactive jobs meant for debugging your code on GPU.

```
qsub -I -l select=1:mem=50GB:ncpus=10:ngpus=1 -l walltime=02:00:00 -q volta_login
```

- -I : Interactive
  - Mem: Max 96GB, Min 50GB
  - ncpus: 6 to 20
  - ngpus: 1 to 2 (Choose less, wait less)
  - -q : volta\_login queue
1. Default Walltime: 1 hour
  2. Max Walltime: 4 hours

# Short Interactive Jobs

- Once the interactive job is launched, you will be brought into **volta01**.
- In Volta01, you can launch a container of your choice to debug your deep learning python script.

```
qsub -I -l select=1:mem=50GB:ncpus=10:ngpus=1 -l walltime=02:00:00 -q volta_login
```

```
singularity exec $image bash
```

- You can replace `$image` with the container image of your choice, e.g.:  
`/app1/common/singularity-img/3.0.0/tensorflow_1.12-nvcr_19.01-py3.simg`

```
[ccekwk@atlas9-c01 horovod]$ qsub -I -l select=1:ncpus=5:ngpus=1 -l walltime=0:05:00 -q volta_login
qsub: waiting for job 789846.venus01 to start
qsub: job 789846.venus01 ready

[ccekwk@volta01 ~]$ uname -a
Linux volta01 3.10.0-862.el7.x86_64 #1 SMP Fri Apr 20 16:44:24 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux
[ccekwk@volta01 ~]$ cat /etc/centos-release
CentOS Linux release 7.5.1804 (Core)
[ccekwk@volta01 ~]$
```

```
[ccekwk@volta01 ~]$ singularity exec /app1/common/singularity-img/3.0.0/tensorflow_1.12_nvcr_19.01-py3.simg bash
ccekwk@volta01:~$ uname -i
x86_64
ccekwk@volta01:~$ cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04.5 LTS"
```

# Interactive Jupyter Notebook

- You can run a Jupyter Notebook instance for a short amount of time (**up to 4 hours**) using the **volta\_login** interactive queue
- The Jupyter Notebook instance will run inside a container of your choice.



# Interactive Jupyter Notebook

- Login to [Atlas9](#) or [Atlas8](#) and set your Jupyter password (ONLY DO ONCE)
- In Atlas9 run the following commands:

```
bash
module load singularity
singularity exec $image jupyter notebook --generate-config
singularity exec $image jupyter notebook password
```

- You can replace `$image` with the container image of your choice, e.g.:  
`/app1/common/singularity-img/3.0.0/tensorflow_1.12-nvcr_19.01-py3.simg`
- Set your password as prompted

# Interactive Jupyter Notebook

- Launch an Interactive Job from Atlas9
- Change walltime and other resources as required

```
qsub -I -l select=1:mem=10GB:ncpus=5:ngpus=1 -q volta_login -l walltime=0:10:00
```

```
[ccekwk@atlas9-c01 ~]$ qsub -I -l select=1:mem=10GB:ncpus=5:ngpus=1 -q volta_login -l walltime=0:10:00
qsub: waiting for job 797088.venus01 to start
qsub: job 797088.venus01 ready
[ccekwk@volta01 ~]$
```

# Interactive Jupyter Notebook

- In Volta1 node, launch jupyter notebook in a container of your choosing
- Execute the following command in 1 line

```
singularity exec $image jupyter notebook --no-browser --port=8889 --ip=0.0.0.0
```

- You can replace \$image with the container image of your choice, e.g.:  
/app1/common/singularity-img/3.0.0/tensorflow\_1.12\_nvcr\_19.01-py3.simg

```
[I 08:39:14.135 NotebookApp] JupyterLab extension loaded from /usr/local/lib/python3.5/dist-packages/jupyterlab
[I 08:39:14.135 NotebookApp] JupyterLab application directory is /usr/local/share/jupyter/lab
[I 08:39:14.136 NotebookApp] Serving notebooks from local directory: /home/svu/ccekwk
[I 08:39:14.136 NotebookApp] The Jupyter Notebook is running at:
[I 08:39:14.136 NotebookApp] http://(volta01 or 127.0.0.1):8889/
[I 08:39:14.136 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```

# Interactive Jupyter Notebook

- On your personal computer in a new Mobaxterm tab run the following command

Entry Port	Target Host	Exit Port	You	Tunnel Host
8888	volta01	8889	nusnet_id@atlas9	

`ssh -L 8888:volta01:8889 nusnet_id@atlas9`

```
[2019-02-27 16:40.26] ~  
[ccekwk.WKEX360] > ssh -L 8888:volta01:8889 atlas9
```

- Open your browser and browse to `http://localhost:8888`

Your Computer	Entry Port
	8888

# Interactive Jupyter Notebook

- When launching jupyter notebook, if you receive an error that says port 8889 is already in use, choose another port

e.g.

- `singularity exec $image jupyter notebook --no-browser --port=8181 --ip=0.0.0.0`
- `ssh -L 8888:volta01:8181 nusnet_id@atlas9`

# Batch Jobs

# Batch Jobs

- Up to 2 GPUs per job
  - Please request only what you need.
  - ENSURE THAT YOUR CODE IS MODEL OR DATA PARALLEL
  - Using >1 GPU has to be **explicitly programmed** in your code.
  - Choose LESS, wait LESS
- Most applications use only 1 GPU by default
  - Requesting >1 GPUs does not make it faster. The other GPUs will **not be utilised** at all.
- Most applications **will not scale** well with > 1 GPUs
- 1 GPU is most likely more than enough as the V100-32GB has large memory capacity

# Queue Resources

Max RAM = 300gb

Max No. of CPU cores = 20

Max No. of GPUs = 2

Max Walltime = 72:00:00

Minimum No. of CPU cores = 5

Minimum No. of GPU = 1

Default Walltime = 04:00:00

Request CPU Core in increments of 1



# Sample Job Script

For Batch Jobs

Note:

1. **Do not copy and paste** the job script in the next slide directly into your text editor.
2. Please **type it out manually** to avoid hidden characters.
3. Ensure that the jobscript **conforms to BASH syntax.**

```
#!/bin/bash
#PBS -P project_name
#PBS -j oe
#PBS -N tensorflow
#PBS -q volta_gpu
#PBS -l select=1:ncpus=10:mem=80gb:ngpus=1
#PBS -l walltime=24:00:00
```

```
cd $PBS_O_WORKDIR;
np=$(cat ${PBS_NODEFILE} | wc -l);
```

```
image="/app1/common/singularity-img/3.0.0/tensorflow_1.12_nvcr_19.01-py3.simg"
```

```
singularity exec $image bash << EOF > stdout.$PBS_JOBID 2> stderr.$PBS_JOBID
```

```
python cifar10_resnet.py
```

```
# you can put more commands here
# echo "Hello World"
```

```
EOF
```

Green is user configurable  
Black is fixed

**Wrong:**

```
image = "/path/to/container/"
```

**Correct**

```
image="/path/to/container"
```

# Multi-GPU Scaling

# Using Multiple GPUs

1. Ensure that your code allows for Multi-GPU training
  - a. [DataParallel](#) on Pytorch
  - b. [Distributed training](#) with Keras
  - c. [Custom training loops \(distributed\)](#) with Tensorflow
  - d. [Horovod](#) for Tensorflow/Keras/Pytorch
2. Then request for 2 GPUs
  - a. `#PBS -l select=1:ncpus=10:mem=80gb:ngpus=2`
  - b. Do NOT request for 2 GPUs if your code does not meet the conditions in Step 1.

# Scaling for Multi GPU with Horovod

For better scaling when utilising multiple gpus, use Horovod.  
Distributed training framework for TensorFlow, Keras, PyTorch, and MXNet.

To use Horovod, you need to set the following in your job script.

```
#PBS -l select=1:ncpus=10:mem=100gb:ngpus=2
```

```
NCCL_DEBUG=INFO ; export NCCL_DEBUG  
mpirun -np 2 -x NCCL_DEBUG python keras_mnist_advanced.py
```

Use mpirun to execute python

```
#!/bin/bash
#PBS -P volta_pilot
#PBS -j oe
#PBS -N tensorflow
#PBS -q volta_gpu
#PBS -l select=1:ncpus=20:mem=100gb:ngpus=2
#PBS -l walltime=24:00:00
```

```
cd $PBS_O_WORKDIR;
np=$(cat ${PBS_NODEFILE} | wc -l);
```

```
image="/app1/common/singularity-img/3.0.0/tensorflow_1.12_nvcr_19.01-py3.simg"
```

```
singularity exec $image bash << EOF > stdout.$PBS_JOBID 2> stderr.$PBS_JOBID
```

```
NCCL_DEBUG=INFO ; export NCCL_DEBUG
mpirun -np 2 -x NCCL_DEBUG python keras_mnist_advanced.py
```

```
EOF
```

Green is user configurable  
Black is fixed

# Multi-Node Multi-GPU Scaling

- Not supported/allowed in NUS HPC Volta Cluster
- Use NSCC DGX-1 Cluster
  - 8 x V100-16GB, 6 Nodes



# Python Packages

# Installing Python Packages: Specify Custom Path (Recommended)

On Atlas8 (NO NEED TO LAUNCH JOB):

```
module load singularity
```

```
singularity exec /app1/common/singularity-img/3.0.0/[yourcontainer.simg] bash
```

Executing the above commands will bring you into the container.

In Container:

```
pip install --prefix=/home/svu/[nusnet_id]/volta_pypkg/ package_name
```

Note: volta\_pypkg is an example directory. You can name it anything you want when you create it.

Replace [nusnet\_id] with YOUR NUSNET ID.

In Host

```
[ccekwk@atlas8-c01 ~]$ module load singularity
Welcome to Singularity, you can find prebuilt images here:
/app1/common/singularity-img/3.0.0
usage:
```

In Container

```
[ccekwk@atlas8-c01 ~]$ singularity exec /app1/common/singularity-img/3.0.0/tensorflow_1.12_nvcr_19.01-py3.simg bash
```

```
ccekwk@atlas8-c01 ~$ pip install --prefix=/home/svu/ccekwk/volta_pypkg/ opencv-python-headless
```

```
Collecting opencv-python-headless
```

```
  Downloading https://files.pythonhosted.org/packages/ee/66/64c85a32937e6bcbf68476ae5bc8fee3bed5596b6681909be7db5640b57c/opencv
```

```
100% | 18.8MB 231kB/s
```

```
Requirement already satisfied: numpy>=1.11.1 in ./local/lib/python3.5/site-packages (from opencv-python-headless) (1.15.1)
```

```
Installing collected packages: opencv-python-headless
```

```
Successfully installed opencv-python-headless-4.0.0.21
```

# Installing Python Packages: Default User Package Path (Not Recommended)

On **Atlas8** (NO NEED TO LAUNCH JOB):

```
$ module load singularity
```

```
$ singularity exec /app1/common/singularity-img/3.0.0/[cont.simg] bash
```

In Container:

```
$ pip install --user package_name
```

Python in the container will automatically locate packages installed with this method

# Pip Install Errors 1

If you encounter this error, it means you're using **ATLAS 9** and not **ATLAS 8**.

Use ATLAS 8 ONLY

```
atlas9-c01:~$ pip install --prefix=/home/svu volta_pypkg/ arlpy
Collecting arlpy
  Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after co
nnection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='files.pythonhosted.org',
port=443): Read timed out. (read timeout=15)",)': /packages/c3/ff/f98fbded05f320c7cd15d43
5bd4c51d2e1e983cbe85fee98de00d2e63581/arlpy-1.6.1.tar.gz
  Retrying (Retry(total=3, connect=None, read=None, redirect=None, status=None)) after co
nnection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='files.pythonhosted.org',
port=443): Read timed out. (read timeout=15)",)': /packages/c3/ff/f98fbded05f320c7cd15d43
5bd4c51d2e1e983cbe85fee98de00d2e63581/arlpy-1.6.1.tar.gz
  Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after co
nnection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='files.pythonhosted.org',
port=443): Read timed out. (read timeout=15)",)': /packages/c3/ff/f98fbded05f320c7cd15d43
5bd4c51d2e1e983cbe85fee98de00d2e63581/arlpy-1.6.1.tar.gz
  Retrying (Retry(total=1, connect=None, read=None, redirect=None, status=None)) after co
nnection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='files.pythonhosted.org',
port=443): Read timed out. (read timeout=15)",)': /packages/c3/ff/f98fbded05f320c7cd15d43
5bd4c51d2e1e983cbe85fee98de00d2e63581/arlpy-1.6.1.tar.gz
  Retrying (Retry(total=0, connect=None, read=None, redirect=None, status=None)) after co
nnection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='files.pythonhosted.org',
port=443): Read timed out. (read timeout=15)",)': /packages/c3/ff/f98fbded05f320c7cd15d43
5bd4c51d2e1e983cbe85fee98de00d2e63581/arlpy-1.6.1.tar.gz
Could not install packages due to an EnvironmentError: HTTPSConnectionPool(host='files.py
thonhosted.org', port=443): Max retries exceeded with url: /packages/c3/ff/f98fbded05f320
c7cd15d435bd4c51d2e1e983cbe85fee98de00d2e63581/arlpy-1.6.1.tar.gz (Caused by ReadTimeou
tError("HTTPSConnectionPool(host='files.pythonhosted.org', port=443): Read timed out. (read
timeout=15)",))

You are using pip version 18.1, however version 19.2.3 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

# Pip Install Errors 2

```
You are using pip version 18.1, however version 19.2.3 is available.  
You should consider upgrading via the 'pip install --upgrade pip' command.
```

This is not an **error**.

Do not attempt to **upgrade** pip.

# Using User-installed Python Packages: Custom Path (Recommended)

In job script add the following:

If you used Tensorflow container:

```
export PYTHONPATH=$PYTHONPATH:/home/svu/[nusnet_id]/volta_pypkg/lib/python3.5/site-packages
```

If you used Pytorch container:

```
export PYTHONPATH=$PYTHONPATH:/home/svu/[nusnet_id]/volta_pypkg/lib/python3.6/site-packages
```

**If you named volta\_pypkg something else, please remember to change it**

Replace [nusnet\_id] with YOUR NUSNET ID

```
ccekwk@atlas8-c01 ~$ PYTHONPATH=$PYTHONPATH:/home/svu/ccekwk/volta_py
volta_py/      volta_pypkg/
ccekwk@atlas8-c01 ~$ PYTHONPATH=$PYTHONPATH:/home/svu/ccekwk/volta_pypkg/
bin/ lib/
ccekwk@atlas8-c01 ~$ PYTHONPATH=$PYTHONPATH:/home/svu/ccekwk/volta_pypkg/lib/python3.5/site-packages/
ccekwk@atlas8-c01 ~$ export PYTHONPATH
ccekwk@atlas8-c01 ~$ python
Python 3.5.2 (default, Nov 12 2018, 13:43:14)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'4.0.0'
>>> 
```



# Batch Job Script for volta\_gpu with Custom Python Lib Path

```
#!/bin/bash
#PBS -P volta_pilot
#PBS -j oe
#PBS -N tensorflow
#PBS -q volta_gpu
#PBS -l select=1:ncpus=5:mem=50gb:ngpus=1
#PBS -l walltime=24:00:00
```

```
cd $PBS_O_WORKDIR;
np=$(cat ${PBS_NODEFILE} | wc -l);
```

```
image="/app1/common/singularity-img/3.0.0/tensorflow_1.12_nvcr_19.01-py3.simg"
```

```
singularity exec $image bash << EOF > stdout.$PBS_JOBID 2> stderr.$PBS_JOBID
```

```
PYTHONPATH=$PYTHONPATH:/home/svu/[nusnet_id]/volta_pypkg/lib/python3.5/site-packages
export PYTHONPATH
```

```
python cifar10_resnet.py
EOF
```

Green is user configurable  
Black is fixed

# Internet Access

# Internet Access from within Containers

If for some reason you need to access the internet from within a container, do the following:

1. Login to Atlas8 and run the following commands
2. `module load singularity`
3. `singularity exec $container_path bash`

Replace `$container_path` with container of your choice.

e.g.: `singularity exec /app1/common/singularity-img/3.0.0/CUDA/cuda_10.0-cudnn7-devel-ubuntu16.04-py3.simg bash`

# PBS Job Scheduler

# Steps

## You have to run:

1. Prepare your **python script** in your working directory
2. **Create a PBS job script** and save it in your working directory
  - a. Example job scripts are in the following 2 slides
3. **Submit PBS job script** to PBS Job Scheduler

## Server will run:

1. Job is in PBS Job Scheduler queue
2. Job Scheduler waits for server resources to be available
3. If available, Job Scheduler runs your script on remote gpu server

# Submitting a Job

Save your job script (previous slides for examples) in a text file (e.g. train.pbs) then run the following commands

```
shell$ qsub train.pbs
675674.venus01
```

```
shell$ qstat -xfn
```

```
venus01:
```

Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Req'd Memory	Req'd Time	S	Elap Time
669468.venus01	ccekkw	volta	cifar_noco	--	1	1	20gb	24:00	F	--
--										
674404.venus01	ccekkw	volta	cifar_noco	--	1	1	20gb	24:00	F	--
TestVM/0										
675674.venus01	ccekkw	volta	cifar_noco	--	1	1	20gb	24:00	Q	--
--										

Statuses: Q(ueue), F(inish), R(unning), E(nding), H(old)

```
[ccekwk@atlas8-c01 classification]$ qsub train.pbs
```

```
697978.venus01
```

```
[ccekwk@atlas8-c01 classification]$ qstat -xfn
```

```
venus01:
```

Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Req'd Memory	Req'd Time	S	Elap Time
695126.venus01	ccekwk	azgpu	cifar_noco	--	1	4	40gb	24:00	F	--
--										
697978.venus01	ccekwk	azgpu	cifar_noco	--	1	4	40gb	24:00	R	--
TestVM/0*4										

```
[ccekwk@atlas8-c01 classification]$ █
```

Statuses: Q(ueue), F(inish), R(unning), E(nding), H(old)

## Useful PBS Commands

Action	Command
Job submission	<code>qsub my_job_script.txt</code>
Job deletion	<code>qdel my_job_id</code>
Job listing (Simple)	<code>qstat</code>
Job listing (Detailed)	<code>qstat -ans1</code>
Queue listing	<code>qstat -q</code>
Completed Job listing	<code>qstat -H</code>
Completed and Current Job listing	<code>qstat -x</code>
Full info of a job	<code>qstat -f job_id</code>



# Log Files

- Output (stdout)
  - `stdout.$PBS_JOBID`
- Error (stderr)
  - `stderr.$PBS_JOBID`
- Job Summary
  - `job_name.o$PBS_JOBID`

```
[ccekwk@atlas8-c01 classification]$ ls -l
total 16604
-rw----- 1 ccekwk admin    15325 Nov 12 12:24 cifar10_resnet.py
-rw----- 1 ccekwk admin     865 Nov 12 12:26 cifar_nocont.o697978
drwx----- 2 ccekwk admin     348 Nov 12 12:28 logs
-rw----- 1 ccekwk admin 13589605 Oct 19 11:04 logs.tar.gz
drwx----- 3 ccekwk admin     456 Sep 21 16:04 mnist
drwxr-xr-x 2 ccekwk admin      98 Nov 12 12:26 saved_models
-rw----- 1 ccekwk admin    1209 Nov 12 12:26 stderr.697978.venus01
-rw----- 1 ccekwk admin   62224 Nov 12 12:26 stdout.697978.venus01
-rw----- 1 ccekwk admin     832 Oct  3 13:29 tf_gcpu24.pbs
-rw----- 1 ccekwk admin     849 Sep 28 16:39 tf.pbs
-rw----- 1 ccekwk admin     612 Oct  1 08:55 train_gpu_container.pbs
-rw----- 1 ccekwk admin     300 Nov  8 13:21 train.pbs
[ccekwk@atlas8-c01 classification]$
```

# Status of Queues

```
[ccekwwk@atlas9-c01 atlas9_test]$ qstat -q

server: venus01

Queue           Memory CPU Time  Walltime Node  Run  Que  Lm  State
-----
login           --    --    720:00:00 --    0    0    --  E R
Q3500           --    --    --         --    35    0    --  E R
Q3800           --    --    720:00:00 --    0    0    --  E R
C2100           --    --    720:00:00 --    1    0    --  E R
Q3300           --    --    --         --    0    0    --  D S
Q3400           --    --    --         --    0    0    --  D R
Q3700           --    --    --         --    0    0    --  E R
Q3600           --    --    --         --    0    0    --  D R
normal          --    --    720:00:00 --    0    0    --  D R
Q3200           --    --    --         --    2    0    --  E R
iworkq          --    --    48:00:00  --    3    12   --  E R
dmloginv4       --    --    24:00:00  --    0    0    --  D R
dmlogina7       --    --    24:00:00  --    0    0    --  E R
benchmark2      --    --    720:00:00 --    0    0    --  E R
dmlogina8       --    --    24:00:00  --    3    0    --  E R
dmlogint2       --    --    24:00:00  --    0    0    --  E R
benchmark       --    --    720:00:00 --    0    0    --  E R
dmlogingold     --    --    24:00:00  --    0    0    --  E R
dmlogina6       --    --    24:00:00  --    2    0    --  E R
dmlogina5       --    --    24:00:00  --    0    1    --  E R
Q3100           --    --    --         --    5    0    --  E R
parallel24      --    --    360:00:00 --    47   58   --  E R
parallel_test   --    --    01:00:00  --    0    0    --  E R
azgpu           --    --    48:00:00  --    0    0    --  E R
openmp          --    --    720:00:00 --    10   14   --  E R
short           --    --    24:00:00  --    1    0    --  E R
serial          --    --    720:00:00 --    83   80   --  E R
gpu             --    --    720:00:00 --    0    0    --  E R
parallel8       --    --    720:00:00 --    37    2    --  E R
parallel12      --    --    720:00:00 --    208  134   --  E R
parallel20      --    --    720:00:00 --    33   11   --  E R
volta_gpu       375gb  --    48:00:00  --    3    0    --  E R
volta_login     96gb   --    04:00:00  --    0    0    --  E R
-----
473    312
```

See statuses of Queue:

- Max Memory
- Max Walltime
- Number of running jobs
- Number of queued jobs

Run `qstat -q` command on login node.

# View Queue Configuration

```
[ccekwk@atlas9-c01 atlas9_test]$ qstat -Qf volta_login
Queue: volta_login
  queue_type = Execution
  total_jobs = 0
  state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:0 Exiting:0 Begun
               :0
  from_route_only = False
  resources_max.mem = 96gb
  resources_max.ncpus = 20
  resources_max.ngpus = 2
  resources_max.walltime = 04:00:00
  resources_min.ncpus = 1
  resources_min.ngpus = 1
  resources_default.walltime = 01:00:00
  default_chunk.model = volta
  default_chunk.ncpus = 1
  default_chunk.ngpus = 1
  resources_available.model = volta
  resources_assigned.mem = 0gb
  resources_assigned.mpi_procs = 0
  resources_assigned.ncpus = 0
  resources_assigned.nodect = 0
  node_group_key = cluster
  enabled = True
  started = True
```

To see individual queue configuration and resource limits use the following command:

```
qstat -Qf queue_name
```

# Status of Nodes

```
[ccekwk@atlas9-c01 atlas9_test]$ gstat
```

```
=====
TIME: Tue Mar  5 11:30:01 SGT 2019
=====
```

## ===== Status of Parallel Queues =====

Queue :	Running :	Waiting
parallel20 :	33 :	11
parallel24 :	47 :	58
parallel12 :	196 :	39
parallel8 :	36 :	3

## === Status of Serial/Openmp/App Queues =====

Queue :	Running :	Waiting
gpu :	0 :	0
openmp :	10 :	28
serial :	83 :	160
short :	1 :	0
volta_gpu :	3 :	0

## ===== Status of all compute nodes =====

Cluster :	Total	Busy	Partial-Free	Free	Off	Down	0th
atlas5 :	63	52	2	1	3	3	2
atlas6 :	96	89	1	3	0	0	3
atlas7 :	152	95	4	40	0	1	11
atlas8 :	64	52	1	10	0	0	1
atlas9 :	23	14	9	0	0	0	0
gold :	15	1	0	12	0	0	2
tiger2 :	96	57	4	33	0	0	2
volta :	5	0	2	3	0	0	0

To view status of nodes use the following command

`gstat`

# Job Chaining and Dependencies

Execute jobs in sequence

- `qsub -W depend=afterok:<Job-ID> <JOB SCRIPT>`
  - `qsub -W depend=afterany:836578.venus01 volta_benchmark.pbs`
- Job script `<QSUB SCRIPT>` will be submitted after the Job, `<Job-ID>` is successfully completed. Useful options to "depend=..." are:
  - **afterok:**`<Job-ID>` Job is scheduled if the Job `<Job-ID>` exits without errors or is successfully completed.
  - **afternotok:**`<Job-ID>` Job is scheduled if the Job `<Job-ID>` exited with errors.
  - **afterany:**`<Job-ID>` Job is scheduled if the Job `<Job-ID>` exits with or without errors

```

ccekwk@atlas9-c01 /hpctmp/ccekwk/tf_benchmarks/container$ qsub volta_benchmark.pbs
836578.venus01
ccekwk@atlas9-c01 /hpctmp/ccekwk/tf_benchmarks/container$ vim volta_benchmark.pbs
ccekwk@atlas9-c01 /hpctmp/ccekwk/tf_benchmarks/container$ qsub -W depend=afterany:836578.venus01 volta_benchmark.pbs
836582.venus01
ccekwk@atlas9-c01 /hpctmp/ccekwk/tf_benchmarks/container$ qstat -fns1

venus01:

```

Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Req'd Memory	Req'd Time	Elap S	Elap Time
836578.venus01	ccekwk	volta_lo	gpu_tf_ben	392016	1	20	80gb	04:00 R	00:01	volta01/0*20
Job run at Thu Apr 18 at 09:00 on (volta01:ncpus=20:mem=83886080kb:ngpu...										
836582.venus01	ccekwk	volta_lo	gpu_tf_ben	--	1	20	80gb	04:00 H	--	--

# Tips and Tricks

# Recommendations

- Max walltime: 72 hours
- Default walltime: 24 hours
- Implement weights/model checkpointing
  - Save your model weights after every training epoch
- Save weights to /hpctmp/nusnet\_id/ directory
  - Backup your important weights from /hpctmp often to your own computer
- Use volta\_login (interactive mode) to test your python scripts before submitting to volta\_gpu



# Speed Ups (Data Access)

Copy **large datasets** or **datasets with a lot of small files** to `/scratch/nusnet_id` on Volta nodes (SSD drives local on each Volta node)

Make directory in `/scratch`: **This creates a directory if it doesn't exist.**

```
mkdir -p /scratch/`whoami`
```

**Sync** dataset folder to `/scratch` using `rsync`: **Synchronises dataset in hpctmp to /scratch folder**

```
rsync -hav /hpctmp/nusnet_id/example/dataset_folder /scratch/nusnet_id/
```

`dataset_folder` will appear in `/scratch/nusnet_id/dataset_folder`

Please put the above commands in your **job script**.

Remember to set the correct dataset path in your python script

## Batch Job Script for volta\_gpu with Speed Ups (Data Access)

```
#!/bin/bash
#PBS -P volta_pilot
#PBS -j oe
#PBS -N tensorflow
#PBS -q volta_gpu
#PBS -l select=1:ncpus=5:mem=50gb:ngpus=1
#PBS -l walltime=24:00:00
```

```
cd $PBS_O_WORKDIR;
np=$(cat ${PBS_NODEFILE} | wc -l);
mkdir -p /scratch/nusnet_id
rsync -hav /hpctmp/nusnet_id/example/dataset_folder /scratch/nusnet_id/
```

```
image="/app1/common/singularity-img/3.0.0/tensorflow_1.12_nvcr_19.01-py3.simg"
```

```
singularity exec $image bash << EOF > stdout.$PBS_JOBID 2> stderr.$PBS_JOBID
PYTHONPATH=$PYTHONPATH:/home/svu/[nusnet_id]/volta_pypkg/lib/python3.5/site-packages
export PYTHONPATH
```

```
python cifar10_resnet.py
EOF
```

Green is user configurable  
Black is fixed

# Speed Ups (Image Processing)

If you are using any image processing that utilises PIL/Pillow,

- install an optimised version called Pillow-SIMD
  - `pip uninstall pillow`
  - `pip install pillow-simd`
- More info here <https://github.com/uploadcare/pillow-simd>

# Tips on How to Game the Queue

1. Request less or reasonable amount of resources (or request just enough).
  - a. Less: e.g. NCPUS or MEM
    - i. Each node only has a maximum of 40 NCPUS and 300GB of RAM shared with 4 GPUs
    - ii. if User A requests for 20 NCPUs, 100GB RAM and 1 GPU. User B requests for 20 NCPUs, 100 GB RAM and 1 GPU, the remaining 2 GPU will not be able to be allocated to other users.
  - b. Reasonable NCPUs or MEM
    - i. 5 to 10, 15 NCPUs is good. Don't request weird numbers like 11, 13, 16, 17, etc
    - ii. 50~80, 90GB of MEM is good. If you need more than 100 GB, you might not be doing Deep Learning efficiently.
2. Load your data in batches, do not load 200GB worth into RAM at the same time.
  - a. It doesn't make sense to load 200GB into RAM at the same time as the GPU only has 32GB of memory.

# Debugging

# Find Out Process Memory Utilisation

Find out how much memory your program/script is using:

- 1) Ensure job is running
- 2) ssh into node that your job is running on
- 3) Run the following command:

```
ps -u [your_nusnet_id] -o user,vsz,%cpu,cmd
```

Output Format:

```
nusnet_id memory_used_(kb) %CPU Process_Name  
nusnet_id 17330140 99.5 vasp_gpu
```

# Debugging

## **For TF Eager, Keras, Pytorch:**

You can place breakpoints using `pdb.set_trace()` at any line in your code. You can then execute further computations, examine the Tensors/Variables and pinpoint the root cause of the error.

## **Python Debugger (PDB) Guide:**

<https://realpython.com/python-debugging-pdb/>

Set Breakpoint



```
for epoch in range(epochs):  
  
    # -----  
    # Train Discriminator  
    # -----  
  
    # Select a random half of images  
    idx = np.random.randint(0, X_train.shape[0], batch_size)  
    imgs = X_train[idx]  
    import pdb; pdb.set_trace()  
    # Sample noise and generate a batch of new images  
    noise = np.random.normal(0, 1, (batch_size, self.latent_dim))  
    gen_imgs = self.generator.predict(noise)
```



# Debugging

## Step 1: Launch Interactive Job

```
ccekwk@atlas9-c01 /hpctmp/ccekwk/volta_scripts/dcgan$ qsub -I -l select=1:mem=40gb:ncpus=10:ngpus=1 -l walltime=02:00:00 -q volta_login
qsub: waiting for job 1056078.venus01 to start
qsub: job 1056078.venus01 ready
```

## Step 2: Launch Container

```
[ccekwk@volta01 ~]$ singularity exec /appl/common/singularity-img/3.0.0/tensorflow_1.13.0-rc0_nvcr_19.02-py3.simg bash
ccekwk@volta01 ~$ ls
```

## Step 3: Run python script

```
ccekwk@volta01 /hpctmp/ccekwk/volta_scripts/dcgan$ python dcgan.py 20
Using TensorFlow backend.
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/op_def_library.py:263: colocate_with (from tensorflow.python
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /home/svu/ccekwk/.local/lib/python3.5/site-packages/keras/backend/tensorflow_backend.py:3445: calling dropout (from tensorflow.
sion.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
2019-07-02 03:07:41.926573: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 2400000000 Hz
2019-07-02 03:07:41.927065: I tensorflow/compiler/xla/service/service.cc:161] XLA service 0x511c960 executing computations on platform Host. Devices:
2019-07-02 03:07:41.927091: I tensorflow/compiler/xla/service/service.cc:168] StreamExecutor device (0): <undefined>, <undefined>
2019-07-02 03:07:42.291926: I tensorflow/compiler/xla/service/service.cc:161] XLA service 0x51f13c0 executing computations on platform CUDA. Devices:
2019-07-02 03:07:42.291962: I tensorflow/compiler/xla/service/service.cc:168] StreamExecutor device (0): Tesla V100-SXM2-32GB, Compute Capability 7.0
2019-07-02 03:07:42.292470: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1433] Found device 0 with properties:
name: Tesla V100-SXM2-32GB major: 7 minor: 0 memoryClockRate(GHz): 1.53
```

## Step 4: PDB Prompt Appears at breakpoint

```
> /hpctmp/ccekwk/volta_scripts/dcgan/dcgan.py(131)train()
-> noise = np.random.normal(0, 1, (batch_size, self.latent_dim))
(Pdb) █
```

```
(Pdb) s
> /hpctmp/ccekwk/volta_scripts/dcgan/dcgan.py(132)train()
-> gen_imgs = self.generator.predict(noise)
(Pdb) p noise
array([[ -0.26828462, -1.13883942, -1.49924144, ...,  1.97896426,
         0.12845006, -0.64945757],
       [  0.55472687,  0.46576234, -0.62202267, ...,  0.50388667,
         0.8577348 ,  0.44606899],
       [ -1.59887642,  0.38554869,  1.28534855, ...,  0.84422676,
         0.40487624,  2.51183261],
       ...,
       [  0.19249113, -1.37917606,  0.88893165, ..., -1.4785005 ,
        -1.596873 , -0.31119037],
       [ -0.59563865,  1.18476751,  0.2268072 , ...,  0.48078546,
        -0.49046079,  0.4200836 ],
       [ -0.54218098, -0.3859126 ,  0.40124725, ...,  0.31483983,
        -0.88258626,  0.59659922]])
(Pdb) █
```

# Debugging

### Code Segment:

```
# Select a random half of images
idx = np.random.randint(0, X_train.shape[0], batch_size)
imgs = X_train[idx]
import pdb; pdb.set_trace()
# Sample noise and generate a batch of new images
noise = np.random.normal(0, 1, (batch_size, self.latent_dim))
gen_imgs = self.generator.predict(noise)
```

## Print Value of Variable

```
(Pdb) p idx
array([53154, 27505, 49633, 15009, 58715, 39393, 25535, 1683, 49882,
       30175, 22606, 36646, 26901, 6541, 31246, 48697, 1580, 58081,
       29218, 41993, 23352, 4250, 32928, 847, 16863, 19970, 33295,
       47031, 46649, 21656, 43015, 14752])
(Pdb) p imgs
array([[[[-1.],
          [-1.],
          [-1.],
          ...,
          [-1.],
          [-1.],
          [-1.]]],
```

## Print Value of Variable

```
(Pdb) p gen_imgs
array([[[[ 0.06812851],
          [ 0.02299314],
          [ 0.0376782 ],
          ...,
          [ 0.01277222],
          [-0.02574343],
          [-0.03162226]],

        [[ 0.08164989],
          [ 0.03416635],
          [ 0.04973982],
          ...,
          [ 0.04154503],
          [ 0.04320211],
          [ 0.0186342 ]],

        [[ 0.04641358],
          [-0.0113236 ],
          [-0.00904338],
```

# Debugging: Commands

Command	Description
p	Print the value of an expression.
pp	Pretty-print the value of an expression.
n	Continue execution until the next line in the current function is reached or it returns.
s	Execute the current line and stop at the first possible occasion (either in a function that is called or in the current function).
c	Continue execution and only stop when a breakpoint is encountered.
unt	Continue execution until the line with a number greater than the current one is reached. With a line number argument, continue execution until a line with a number greater or equal to that is reached.
l	List source code for the current file. Without arguments, list 11 lines around the current line or continue the previous listing.
ll	List the whole source code for the current function or frame.
b	With no arguments, list all breaks. With a line number argument, set a breakpoint at this line in the current file.

# Debugging: Commands

Command	Description
w	Print a stack trace, with the most recent frame at the bottom. An arrow indicates the current frame, which determines the context of most commands.
u	Move the current frame count (default one) levels up in the stack trace (to an older frame).
d	Move the current frame count (default one) levels down in the stack trace (to a newer frame).
h	See a list of available commands.
q	Quit the debugger and exit.

# FAQ/Common Problems

# FAQ

Q: I need to use Keras, how do I install Keras in the container?

A:

1. Use Tensorflow built-in Keras via `import tf.keras`

# FAQ

Q: I followed the guide but Keras is not using GPU!

A:

Check: If you accidentally installed tensorflow. You DO NOT have to install tensorflow as its provided in the container.

```
$ ls ~/.local/lib/python3.5/site-packages/
```

If tensorflow exists in that folder, remove it.

```
pip uninstall tensorflow
```



# FAQ

Q: I pip installed opencv-python but import cv2 gives this error:

```
ImportError: libSM.so.6: cannot open shared object file: No such file or directory
```

A:

```
pip uninstall opencv-python
```

```
pip install --user opencv-python-headless opencv-contrib-python-headless
```

# FAQ

Q: I submitted a job and it failed. The error is

```
#PBS: bad interpreter: No such file or directory
```

A: There are hidden characters (^M, etc) in your job script.

**Check:** `cat -v my_text_file.txt`

**Fix:** Use dos2unix tool to remove them, or manually remove them in vim.

This happens when you create or copy text files from Windows systems to Linux.

# FAQ

Q: I encounter some tkinter error when using matplotlib, I need to install python-tk

A: tkinter is one of many backends for matplotlib. Tkinter is a GUI framework but our environment is headless.

Tkinter will not work. You'd have to use an alternative backend.

```
import matplotlib
matplotlib.use('agg')
import matplotlib.pyplot as plt
```

# FAQ

Q: cv2.imshow() throws an error when i use it in Jupyter notebook.

A:

```
%matplotlib inline
import matplotlib.pyplot as plt
# load image using cv2....and do processing.
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
# as opencv loads in BGR format by default, we want to show it in RGB.
plt.show()
```

# FAQ

Q: I encounter this message in my failed job:

Usage:

```
singularity [global options...] exec [exec options...] <container> <command>
```

A: Please follow the example jobscripts very closely. Ensure that your jobscript conforms to bash syntax.

This error typically occurs when variable assignment is done this way:

```
image = "/path/to/container/" instead of  
image="/path/to/container"
```

# FAQ

Q: I created a conda environment with miniconda and try to run it in my container but it did not work

A: Miniconda is not supported for Deep Learning with containers.

# Additional Support

## nTouch

<https://ntouch.nus.edu.sg/ux/myitapp/#/catalog/home>

# Project Collaboration/ Long Term Engagement or Support

## Email

[dataengineering@nus.edu.sg](mailto:dataengineering@nus.edu.sg)

# nTouch Support Request Template

1. Login node used:
2. Container used:
3. What are you trying to do:
4. Description of error:
5. Screenshots or copy paste log:
6. Copy and paste the following line:
  - a. "I have read the entire Volta Guide and have not found a solution to my problem in the guide."
7. Attach Job Script (if applicable)



Not Supported:  
Conda with Containers

# Installing Miniconda with your Container

## Installing miniconda in a plain container

```
(/app1/common/singularity-img/3.0.0/CUDA/cuda_10.0-cudnn7-devel-ubuntu16.04.simg)
```

### On Atlas 8:

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86\_64.sh  
module load singularity  
singularity exec /app1/common/singularity-img/3.0.0/CUDA/cuda_10.0-cudnn7-devel-ubuntu16.04.simg bash  
bash Miniconda3-latest-Linux-x86\_64.sh
```

#Take note of installation directory target. Change `/my/install/directory` accordingly

```
eval "$( /my/install/directory/miniconda3/bin/conda shell.bash hook )"
```

# run the above command every time before you use conda in a new session.

# You are now ready to run anaconda

# Creating Environment and Installing Packages

Assuming you've followed the previous slide and still on the same session:

```
# Create your environment
```

```
conda create -n my_environment_name python=3.6
```

```
# Activate your environment
```

```
conda activate my_environment_name
```

```
# Install a package
```

```
conda install tensorflow-gpu
```

# Using Conda on Volta

On Volta, in a new session

```
# Load conda
eval "$(/my/install/directory/miniconda3/bin/conda shell.bash hook)"

# Activate your environment
conda activate my_environment_name
```