Using Enki HPC for AI/ML using Python/Keras/TensorFlow and GPUs

Date: Friday, May 29, 2020

Contents

[Enki Architeture 1](#_Toc17710209)

[Cluster Layout 1](#_Toc17710210)

[Head Node Configuration 2](#_Toc17710211)

[Compute Node Configuration 2](#_Toc17710212)

[Enki Computing 2](#_Toc17710213)

[Establish a ssh connection to Enki 2](#_Toc17710214)

[SLURM 5](#_Toc17710215)

[Loading files onto enki. 8](#_Toc17710216)

[Loading AI/ML Keras Tensor Flow modules 12](#_Toc17710217)

[TensorFlow 12](#_Toc17710218)

[Keras Autokin Enki Shell Script 15](#_Toc17710219)

[Running Script on Enki Interactively 17](#_Toc17710220)

[Running Script on Enki Batch 18](#_Toc17710221)

[Related 18](#_Toc17710222)

[Vim Notes 18](#_Toc17710223)

[What is Distributed Deep Learning? 21](#_Toc17710224)

[Anaconda Python Package Environment 21](#_Toc17710225)

# Enki Architeture

## Cluster Layout

The Enki system comprises a head node and 13 compute nodes. Each of the 14 machines runs [Red Hat Enterprise Linux 7](https://en.wikipedia.org/wiki/Red_Hat_Enterprise_Linux#RHEL_7).

## Head Node Configuration

The head node, enki.nist.gov, is accessible via [SSH](https://en.wikipedia.org/wiki/Comparison_of_SSH_clients) from any machine on the NIST intranet. It is equipped with 128 GB DDR4 and two [Intel Xeon Silver 4110](https://ark.intel.com/products/123547/Intel-Xeon-Silver-4110-Processor-11M-Cache-2_10-GHz) CPUs, each with 8 cores (16 threads) clocked at 2.1 to 3.0 GHz. It is not equipped with a [GPGPU](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units). The Enki head node is pre-installed with git, rsync, and other utilities to help import and manage your data.

## Compute Node Configuration

The compute nodes, enki[01-13], are accessible via [SSH](https://en.wikipedia.org/wiki/OpenSSH) from the head node only. Each compute node is an [IBM Power System AC922 "Witherspoon"](https://www.ibm.com/us-en/marketplace/power-systems-ac922) HPC server equipped with 575 GB DDR4 memory and two [IBM POWER9 SMT4 Monza](https://www-355.ibm.com/systems/power/openpower/tgcmDocumentRepository.xhtml?aliasId=POWER9_Monza) CPUs, each with 20 cores (80 threads) clocked at 2.25 GHz.

N.B.: these are [*64-bit ppc64le*](https://en.wikipedia.org/wiki/Ppc64) CPUs capable of 1, 2, or 4-way [*SMT*](https://en.wikipedia.org/wiki/Simultaneous_multithreading), currently configured for 4-way SMT.

Each Witherspoon node is equipped with four [Nvidia Tesla V100 SXM2](https://www.nvidia.com/en-us/data-center/tesla-v100/) GPUs, interconnected with the CPU and other GPUs via [NVLink2](https://www.nvidia.com/en-us/data-center/nvlink/) with 150 GB/s bandwidth per direction (in + out, 300 GB/s total per device). Each V100 has 16 GB HBM2 memory, 5120 CUDA cores, and 640 tensor cores, drawing 300 W under full load.

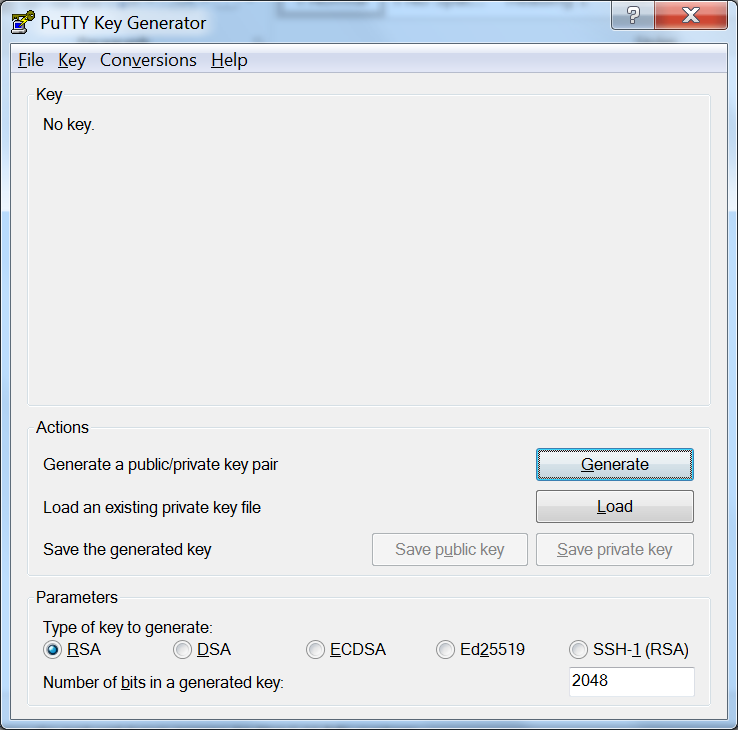
# Enki Computing

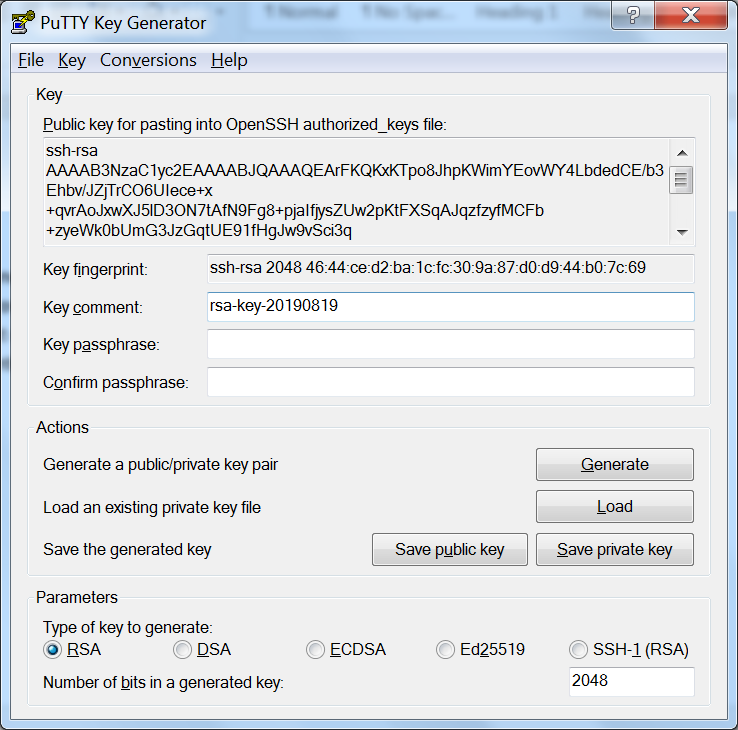
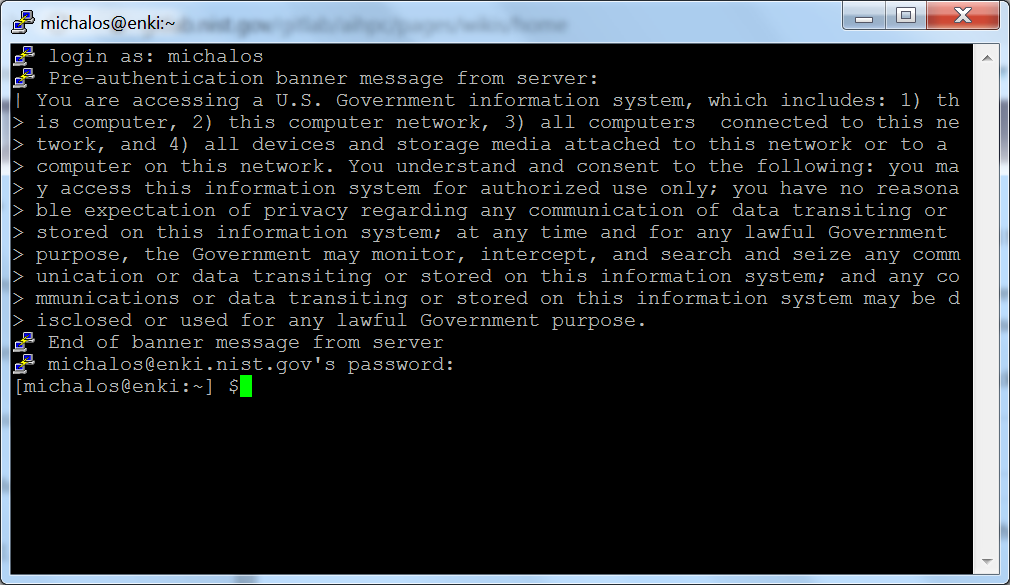
<https://tkphd.ipages.nist.gov/2018-10-03-cuda/> has user guide to operations. Following are my notes.

## Establish a ssh connection to Enki

For windows: (Please note Windows 10 now has openssh, but this uses putty).

1. Install putty: <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>
2. Change directory to C:\Program Files\PuTTY and then shift->right click open shell. To generate an SSH public/private key pair in Windows, simply run "puttygen" in the DOS console or Powershell and this window should pop up:



1. You should generate an RSA type-2 key, at least 1024 bits in size, and preferably larger. You should also provide a passphrase for your key -- passphrases should be at least 16 characters long, and must contain digits, letters, and punctuation. Unlike passwords, passphrases are allowed to contain spaces and dictionary words.  
   
2. Installation consists of simply adding this key to the ~/.ssh/authorized\_keys file on your CTCMS account, and does not require administrative privileges.  Of course, the admin may need to do it for you if you do not yet have access to the CTCMS system.
3. To connect, the recommended procedure is to first run "pageant" (again, probably from the "puTTY" submenu in the "all programs" menu from the "Start" button).  When you first start "pageant", it runs in the background, and it may look like nothing happened.
4. To add your private key to your agent, you can right-click on the pageant icon in the system tray on the bottom right-hand side of the screen, and select "Add".  The agent will present you with a file selector, which you should use to navigate to the ".ppk" file.  When adding the key, the agent will prompt you for the passphrase.
5. To actually connect, you can now open the "puTTY" application itself.  Basic connection is very simple -- on the default page of the puTTY window, there is a blank where you can type the name of your destination machine.  Enter enki.nist.gov, and click "connect".
6. The head node, enki.nist.gov, is accessible via the puTTY SSH from any machine on the NIST intranet. You must have registered, and once registered it uses your NIST credentials for login authentication. Enki is a 13-node IBM Power9 system where each node is equipped with 128 GB DDR4 and two Intel Xeon Silver 4110 CPUs, each with 8 cores (16 threads) clocked at 2.1 to 3.0 GHz. It is not equipped with a GPGPU (from Wikipedia: General-purpose computing on graphics processing units (GPGPU, rarely GPGP) is the use of a graphics processing unit (GPU), which typically handles computation only for computer graphics, to perform computation in applications traditionally handled by the central processing unit (CPU).)
7. A dos command shell will pop up. Put in your NIST credentials if you have already signed up for enki HPCC use.  
     
     
   Now some simple Linux commands to see basic functionality:

michalos@enki:~] $pwd

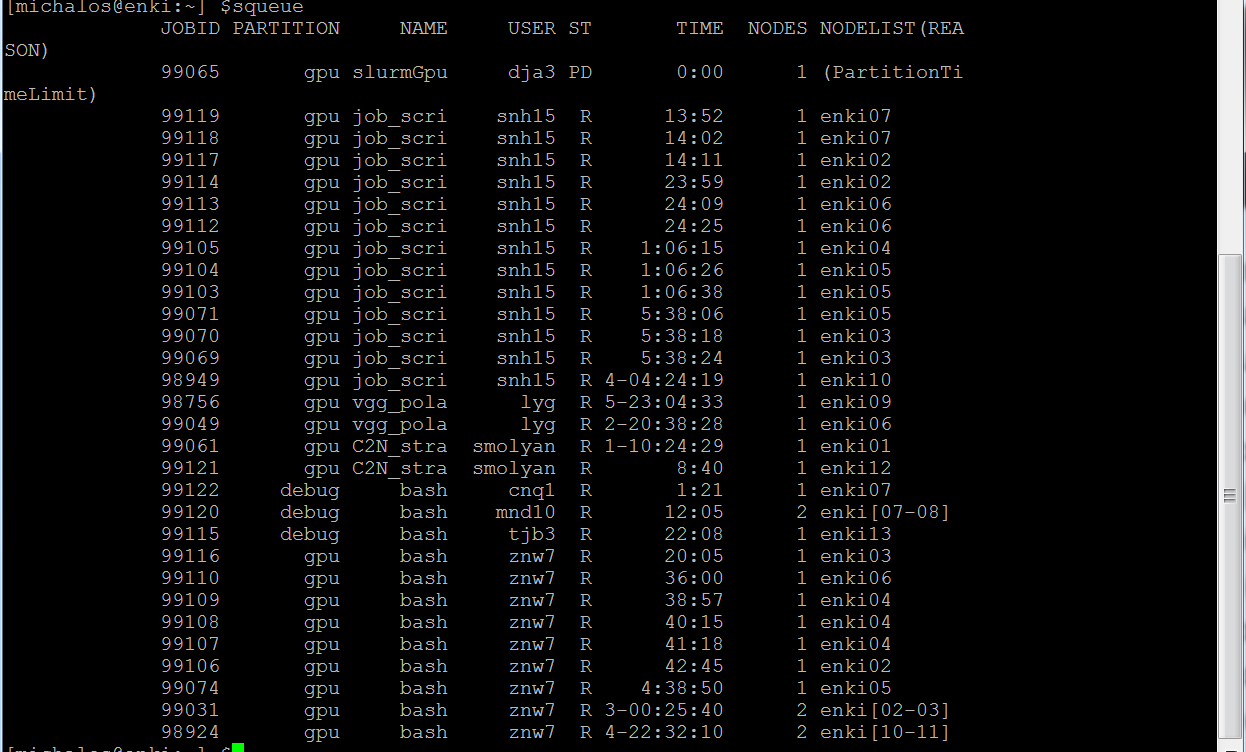
/home/michalos

[michalos@enki:~] $ls -a

. .. .bash\_logout .bash\_profile .bashrc .cache .config .emacs .kshrc .mozilla .ssh .zshrc

## SLURM

Next will be a synopsis of tutorial of the SLURM system found at: <https://slurm.schedmd.com/> Slurm is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters.  
SLURM has a set of commands on the linux path: scontrol, sinfo, squeue, etc. sinfo provides general system state. The squeue command determines what jobs exist on the system for example shown below.



You can start interactive sessions with srun --pty, and batch sessions with sbatch. Use --gres=gpu:# to request GPUs. For example, to get all four GPUs and one CPU core on a node (a single core is the default). The following example requests one hour, zero minutes, and zero seconds in the highest-priority "debug" partition:

[michalos@enki]$ srun --pty --partition=debug --time=1:00:00 --gres=gpu:1 bash

Sbatch is usually included in a bash shell script and starts an Enki batch program.

There are many more options and details on SLURM can be found: <https://gitlab.nist.gov/gitlab/aihpc/pages/wikis/Slurm-Instructions>

1. Enki specific scripting notes
   1. If you're writing an sbatch script, make sure you set the correct working directory using -D. For best performance, this would be a path on /scratch, the node-local SSD.  
      Neither /wrk nor the nodes' /scratch is backed up: please transfer your research results out of Enki as soon as possible.
   2. The maximum runtime on Enki is 14 days, and it is also the default. This is configured as a courtesy, to allow long tasks to run without interruption.
   3. Cross-Compiling Will Not Work Code compiled on enki will not execute on enki[01-13]! This is because the enki instruction set (x86-64) is vastly different from the enki[01-13]s (ppc64le). From <https://gitlab.nist.gov/gitlab/aihpc/pages/wikis/Cluster-Layout>.
   4. Storage:   
       /home/$USER is shared between nodes and backed up. 20 TB shared amongst all users, so don't store anything big. Use this to store programs. This is the slowest storage option.

 /wrk is shared between nodes and not backed up. 1700 TB shared amongst all users. Create /wrk/$USER and use this for your setting up the working data needed for your program, and storing large results. It is faster than /home but slower than /scratch. Some of this will be used for common machine learning data sets (work-in-progress).

[michalos@enki:~] $mkdir /wrk/michalos

mkdir: cannot create directory ‘/wrk/michalos’: File exists

 /scratch is private to each node and not backed up. 5.9 TB per node. This is very fast storage you can use for saving and reloading intermediate results. Put data in /scratch/$USER/$SLURM\_JOB\_ID to avoid collisions. Be nice and clean up after your job is done so other jobs can use it.

1. Slurm Flags

This list should serve as "Cliff's Notes" for the rest of the page.

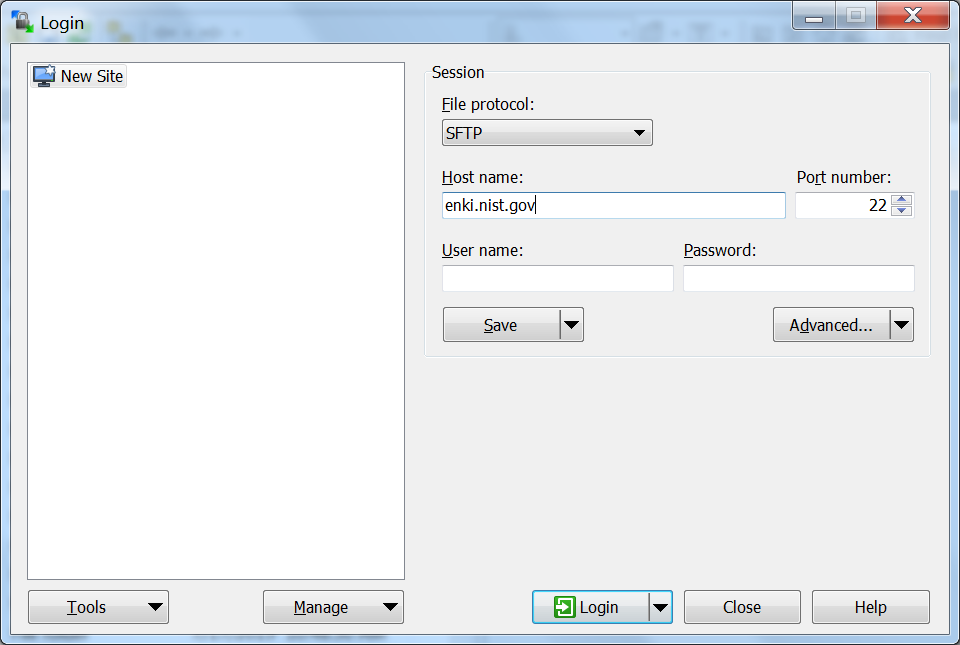
* **--chdir, -D** change directory: should be on /wrk or /scratch. WHY not HOME?
* **--cpus-per-task, -c** number of CPUs (logical cores) required per MPI task (process). Should generally be equal to the number of threads per task: default is -c 1, max is 160.
* **--gres** number of accelerators: default is --gres=gpu:0, max is gpu:4.
* **--nodes, -N** required number of nodes: default is -N 1, max is 13.
* **--ntask, -n** total number of MPI tasks (processes): default is -n 1.
* **--pty** interactive mode: default is *off*
* **--time, -t** maximum runtime (d-hh:mm:ss): default is -t 14-00:00:00, max is 14-00:00:00 (two weeks).
* **--ntasks-per-core=1** indicates to OpenMPI that each task should run on an separate core; without this then OpenMPI gets confused and binds everything to one thread.
* **--exclusive** indicates that the node should not be shared with other running jobs; **--exclusive=user** indicates that it should not be shared with other users.
* **-w enki##** runs on node ##

## Loading files onto enki.

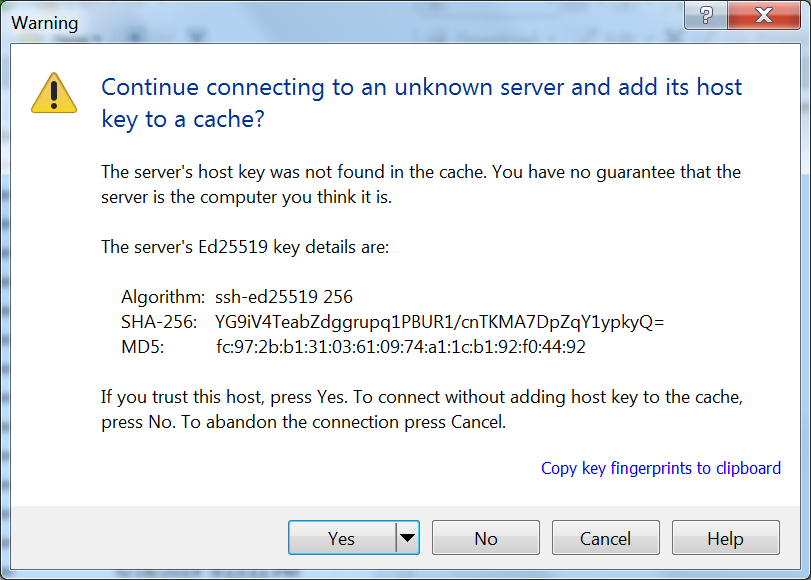
We will use winscp although there are other options. NOTE there is a problem with carriage return/line feed convention between windows and linux. dos2unix is not included in the red hat distribution. I ended up using vim and the command. I used vim and the vim command :set fileformat=unix It seems to work, as things bomb indiscriminantly with the wrong end of line convention.

Download winscp from <https://winscp.net/eng/download.php> and install and run.

A split screen WINSCP dialog will pop up and you will need to log in.

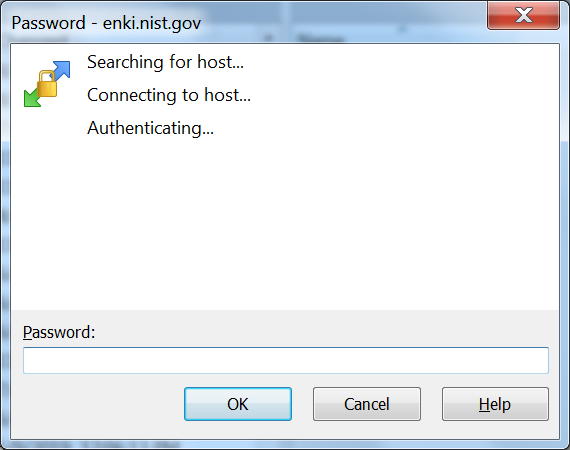


This assumes you have already logged in using putty and put your ssh file in $HOME/.ssh.

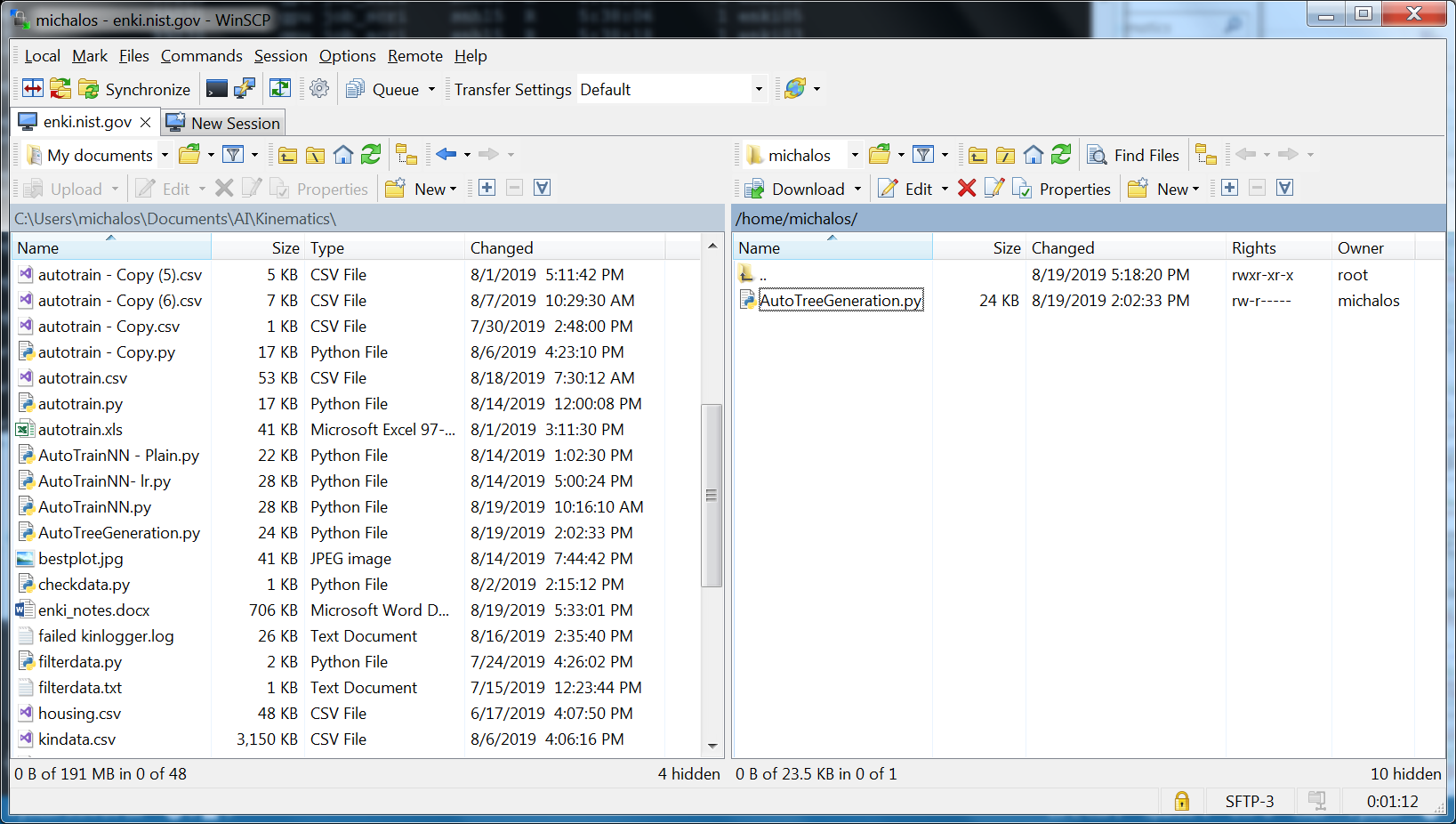


[TODO PUT IN IMAGE OF LOGIN DIALOG]



  
Put in your NIST gene login password.

Then just select the files you want to transfer and then drag over to the Enki sie.



Instead of using winscp you can use github to clone or pull latest version of a project. My project was not big enough to worry about this.

## Loading AI/ML Keras Tensor Flow modules

Following the instructions found at: <https://gitlab.nist.gov/gitlab/aihpc/pages/wikis/Deep-Learning>

TensorFlow, PyTorch, and Caffe2 are included via [Watson Machine Learning Community Edition (formerly PowerAI)](https://developer.ibm.com/linuxonpower/deep-learning-powerai/releases/). The Watson Machine Learn Community Edition (WML CE) packages are distributed as Conda packages in an online Conda repository. Package dependencies are automatically resolved and CUDA, cuDNN, and NCCL are packaged as Conda packages installed by WML CE.

Since our AI/ML program uses Keras with a tensorflow backend, we will investigate the required steps.

## TensorFlow

Note! IBM has deprecated. IBM PowerAI ML/DL packages including tensorflow. From an interactive session (described below) is the output from use the "module load" command:

[michalos@enki13:~/autokin] $module load powerAI/tensorflow

DEPRECATION: this module powerAI/tensorflow has been deprecated. IBM PowerAI ML/DL packages are now distributed as Conda packages. See https://www.ibm.com/support/knowledgecenter/SS5SF7\_1.6.0/navigation/pai\_install.htm for more details.

As a result, only Anaconda/conda instructions are provided to set up the gpu environment.

To run TensorFlow first create an interactive enki job session, set up the Anaconda environment properly, and then create an Anaconda environment for your TensorFlow application, you use conda create to make an Anaconda environment and MUST use conda to install the appropriate powerpc python modules. Below is a ssh session performing these tasks:

# start up an interactive session

[michalos@enki]$ srun -p debug -t 60:00 -n 1 --pty bash

[michalos@node]$ source /opt/anaconda3/etc/profile.d/conda.sh

[michalos@node]$ conda config --prepend channels \

https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda/

[michalos@node]$ mkdir envs

[michalos@node]$ conda create --prefix envs/autokin python=3.6

[michalos@node]$ conda activate /home/michalos/envs/autokin

First, create an interactive job session into the powerpc world

srun -p debug -t 60:00 -n 1 --pty bash

Then the conda environment must be set up properly with the following commd:

> source /opt/anaconda3/etc/profile.d/conda.sh

Conda then must be configured to give priority to installing packages from the IBM channel. Add the PowerAI channel to your Conda configuration by running the following command:

> conda config --prepend channels \

https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda/

Then a user environment must be established for conda where the only valid Python versions with WML CE are Python 2.7 and 3.6.

[michalos@node]$ mkdir envs

[michalos@node]$ conda create --prefix envs/autokin python=3.6

[michalos@node]$ conda activate /home/michalos/envs/autokin

Check that conda python environment was created:

[michalos@enki:~] $conda info --envs

# conda environments:

#

/home/michalos/envs/autokin

base \* /opt/anaconda3

Now we would like to install tensorflow-gpu into our conda environment. We would like to get the latest, which can be done using pip (i.e. pip search tensorflow) however pip will get non-powerpc module builds, AND WE ONLY WANT POWERPC. So we need to use conda.

[michalos@node]$ conda install tensorflow-gpu

You should get the following output for the loaded python modules:

The following NEW packages will be INSTALLED:

\_tflow\_select ibmdl/export/pub/software/server/ibm-ai/conda/linux-ppc64le::\_tflow\_select-2.1.0-gpu\_723.g2e92f34

absl-py pkgs/main/linux-ppc64le::absl-py-0.7.1-py36\_0

astor pkgs/main/linux-ppc64le::astor-0.7.1-py36\_0

c-ares pkgs/main/linux-ppc64le::c-ares-1.15.0-h7b6447c\_1001

cudatoolkit ibmdl/export/pub/software/server/ibm-ai/conda/linux-ppc64le::cudatoolkit-10.1.168-533.g8d035fd

cudnn ibmdl/export/pub/software/server/ibm-ai/conda/linux-ppc64le::cudnn-7.5.1\_10.1-507.gcdf2330

gast pkgs/main/linux-ppc64le::gast-0.2.2-py36\_0

google-pasta ibmdl/export/pub/software/server/ibm-ai/conda/linux-ppc64le::google-pasta-0.1.6-py36\_482.gde1dc70

graphsurgeon ibmdl/export/pub/software/server/ibm-ai/conda/linux-ppc64le::graphsurgeon-0.4.1-py36\_515.gdfe9810

grpcio pkgs/main/linux-ppc64le::grpcio-1.16.1-py36hf8bcb03\_1

nccl ibmdl/export/pub/software/server/ibm-ai/conda/linux-ppc64le::nccl-2.4.7-501.g7a305d1

powerai-license ibmdl/export/pub/software/server/ibm-ai/conda/linux-ppc64le::powerai-license-1.6.1-627.gc80b92e

tensorboard ibmdl/export/pub/software/server/ibm-ai/conda/linux-ppc64le::tensorboard-1.14.0-py36\_d2a1b9e\_3334.gb94cdc6

tensorflow ibmdl/export/pub/software/server/ibm-ai/conda/linux-ppc64le::tensorflow-1.14.0-gpu\_py36\_733.gb94cdc6

tensorflow-base ibmdl/export/pub/software/server/ibm-ai/conda/linux-ppc64le::tensorflow-base-1.14.0-gpu\_py36\_87989f6\_57051.gd288838

tensorflow-estima~ ibmdl/export/pub/software/server/ibm-ai/conda/linux-ppc64le::tensorflow-estimator-1.14.0-py36\_a031b45\_1136.gb94cdc6

tensorflow-gpu ibmdl/export/pub/software/server/ibm-ai/conda/linux-ppc64le::tensorflow-gpu-1.14.0-733.gb94cdc6

tensorrt ibmdl/export/pub/software/server/ibm-ai/conda/linux-ppc64le::tensorrt-5.1.3.6-py36\_515.gdfe9810

termcolor pkgs/main/linux-ppc64le::termcolor-1.1.0-py36\_1

uff ibmdl/export/pub/software/server/ibm-ai/conda/linux-ppc64le::uff-0.6.3-py36\_515.gdfe9810

wrapt pkgs/main/linux-ppc64le::wrapt-1.10.11-py36h14c3975\_2

Now we will try an interactive python to test to conda environment (^D)Log off and log back on to make sure .bashrc changes are working:

[michalos@enki]$ srun -p debug -t 60:00 -n 1 --pty bash

[michalos@enki07:~] $source /opt/anaconda3/etc/profile.d/conda.sh

[michalos@enki07:~] $conda config --prepend channels \

> https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda/

[michalos@enki07:~] $conda activate /home/michalos/envs/autokin

(autokin) [michalos@enki07:~] $python3

Python 3.6.9 |Anaconda, Inc.| (default, Jul 30 2019, 19:18:58)

[GCC 7.3.0] on linux

Type "help", "copyright", "credits" or "license" for more information.

>>> import tensorflow as tf; print(tf.\_\_version\_\_)

1.2.1

>>> (^D)

Now try a test sbatch file tf.slurm whose code is:

#!/bin/bash

#SBATCH -J tensorflow-test

#SBATCH -p gpu

#SBATCH --mail-type=end

#SBATCH --mail-user=john.michaloski@nist.gov

#SBATCH -o %j.out

#SBATCH -e %j.err

#SBATCH -n 1

source /opt/anaconda3/etc/profile.d/conda.sh

conda activate /home/michalos/envs/autokin

python -c 'import tensorflow as tf; print(tf.\_\_version\_\_)

(autokin) [michalos@enki07:~] $ chmod +x tf.slurm

(autokin) [michalos@enki07:~] $ ./tf.slurm

1.2.1

(autokin) [michalos@enki07:~] $

You can monitor your job’s status via squeue. So bring up another putty ssh and see the status.

[michalos@enki:~] $squeue -u michalos

JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)

99201 debug bash michalos R 45:47 1 enki07

This is the interactive debug session.

# Keras Autokin Enki Shell Script

Based on the Enki shell script found at <https://gitlab.nist.gov/gitlab/mmajursk/Object-Detection/blob/master/Yolo_v3/launch_train_sbatch.sh> a shell script file incorporating sbatch was coded.

#!/usr/bin/bash

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# MODIFY THESE OPTIONS

#SBATCH --partition=gpu

#SBATCH --nodes=1

#SBATCH --cpus-per-task=160

#SBATCH --gres=gpu:4

#SBATCH --job-name=y3

#SBATCH --time=72:0:0

#SBATCH -o y3-%N.%j.out

#########################################

# Name of experiment

experiment\_name="autokin-$(date +%Y%m%dT%H%M%S)"

echo "Experiment: $experiment\_name"

#########################################

# Define the term handler function

# note that this is not executed here, but rather

# when the associated signal is sent

term\_handler()

{

echo "function term\_handler called. Cleaning up and Exiting"

# Do nothing

exit -1

}

# associate the function "term\_handler" with the TERM signal

trap 'term\_handler' TERM

#########################################

# FOLDERS

input\_data\_directory="/wrk/michalos/autokin"

output\_directory="/wrk/michalos/autokin"

scratch\_dir="/scratch/${SLURM\_JOB\_ID}"

results\_dir="$output\_directory/$experiment\_name"

mkdir -p ${results\_dir}

mkdir -p "$results\_dir/src"

cp -r . "$results\_dir/src"

echo "Input Directory: $input\_data\_directory"

echo "Output Directory: $output\_directory"

echo "Results Directory: $results\_dir"

echo "Scratch Directory: $scratch\_dir"

echo "Working directory contains: "

ls ${scratch\_dir}

#########################################

# DATA

data\_file="kindata.csv"

# copy data to node

echo "Copying data to Node"

cp -r ${input\_data\_directory}/${data\_file} ${scratch\_dir}/${data\_file}

echo "data copy to node complete"

#########################################

# job configuration

#########################################

# set conda environment - this should have all the imports

source /opt/anaconda3/etc/profile.d/conda.sh

conda activate /home/michalos/envs/autokin

conda config --prepend channels \

https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda/

conda activate /home/michalos/envs/autokin

#########################################

# launch training script with required options

echo "Launching simple automated variable tuning Script"

python3 AutoTreeGeneration.py --dataset="$scratch\_dir/$data\_file" --output\_dir="$results\_dir" | tee "$results\_dir/log.txt"

echo "Job completed"

## Running Script on Enki Interactively

Log into enki using putty ssh as described earlier. Then, log into cluster using srun:

srun -p debug -t 12:00:00 -n 1 --pty bash # 1/2 day

Or miss piggy:

srun --partition=debug --time 3-00:00:00 --cpus-per-task 160 --gres=gpu:4 --pty bash # 3 days

Make sure the script is executable

[michalos@enki05:~/autokin] $chmod +x launch\_autokin\_sbatch.sh

run shell script

[michalos@enki05:~/autokin] $./launch\_autokin\_sbatch.sh

Experiment: autokin-20190821T114826

Input Directory: /wrk/michalos/autokin

Output Directory: /wrk/michalos/autokin

Results Directory: /wrk/michalos/autokin/autokin-20190821T114826

Scratch Directory: /scratch/99271

Working directory contains:

Copying data to Node

data copy to node complete

Warning: 'https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda/' already in 'channels' list, moving to the top

Launching simple automated variable tuning Script

Traceback (most recent call last):

File "AutoTreeGeneration.py", line 6, in <module>

import matplotlib.pyplot as plt

ModuleNotFoundError: No module named 'matplotlib'

Job completed

In total, was missing matplotlib and pandas and scikit-learn (sklearn), so these python modules were added to the autokin conda environment, which caused downgrading of several python packages/modules.

> source /opt/anaconda3/etc/profile.d/conda.sh

[michalos@enki05:~/autokin] conda config --prepend channels\

<https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda/>

[michalos@enki05:~/autokin] $ conda install matplotlib pandas scikit-learn

It appears as if Keras module install covered the other required python modules, e.g., tensorflow etc.

Now run the shell script interactively:

./launch\_autokin\_sbatch.sh

## Running Script on Enki Batch

Once you have the file prepared, you can submit it by:

[user@enki]$ sbatch file

The output will be a job number assigned by the queuing system. The job will be dispatched once sufficient resources are available. Unless otherwise redirected, standard output from the job will appear in a file named "slurm-<job#>.out" in the directory where the "sbatch" command was issued.

sbatch launch\_autokin\_sbatch.sh

The output will be a job number assigned by the queuing system. The job will be dispatched once sufficient resources are available. Unless otherwise redirected, standard output from the job will appear in a file named "slurm-<job#>.out" in the directory where the "sbatch" command was issued.

The normal method to kill a Slurm job is:

scancel <jobid>

Do it twice, check that your job is no longer in squeue.

# Related

## Vim Notes

Editing was done with vim, as emacs

Most macro commands use a prefix ":". To get to command bar hit ":" and then here are some basics:

INSERT MODE (hit Esc to retract)

:w (write file)

:q (quit)

:q! (abort quit – no saves, no questions,)

:set fileformat=unix (convert dos file to unix)

:%s/\r+//g (convert dos file to unix)

Of note, the conversion from DOS/WINDOWS to Linux line feed conversion seems to be best handled in vim by:

:set fileformat=unix (convert dos file to unix)

And using vim from the command line:

vim file.txt -c "set ff=unix" -c ":wq"

And as a pipe command

find . -name '\*.py' -exec vim {} -c "set ff=unix" -c ":wq" \;

Don't do this in your home directory or you will change every .conda/\*.py file….

(autokin) [michalos@enki05:~/autokin] $conda list

# packages in environment at /home/michalos/envs/autokin:

#

# Name Version Build Channel

\_libgcc\_mutex 0.1 main

backports 1.0 py\_2

backports.weakref 1.0rc1 py36\_0

blas 1.0 openblas

bleach 1.5.0 py36\_0

ca-certificates 2019.5.15 1

certifi 2019.6.16 py36\_1

cycler 0.10.0 py36\_0

freetype 2.9.1 h8a8886c\_0

h5py 2.8.0 py36h8d01980\_0

hdf5 1.10.2 hba1933b\_1

html5lib 0.9999999 py36\_0

icu 58.2 h64fc554\_1

keras 2.2.2 0

keras-applications 1.0.4 py36\_1

keras-base 2.2.2 py36\_0

keras-preprocessing 1.0.2 py36\_1

kiwisolver 1.1.0 py36he6710b0\_0

libedit 3.1.20181209 hc058e9b\_0

libffi 3.2.1 hf62a594\_5

libgcc-ng 8.2.0 h822a55f\_1

libgfortran-ng 7.3.0 h822a55f\_1

libopenblas 0.3.6 h5a2b251\_1

libpng 1.6.37 hbc83047\_0

libprotobuf 3.8.0 hd408876\_0

libstdcxx-ng 8.2.0 h822a55f\_1

markdown 3.1.1 py36\_0

matplotlib 3.1.0 py36h5429711\_0

ncurses 6.1 he6710b0\_1

numpy 1.16.4 py36h99e49ec\_0

numpy-base 1.16.4 py36h2f8d375\_0

openssl 1.1.1c h7b6447c\_1

pandas 0.25.0 py36he6710b0\_0

pip 19.2.2 py36\_0

protobuf 3.8.0 py36he6710b0\_0

pyparsing 2.4.2 py\_0

python 3.6.9 h2bede3c\_0

python-dateutil 2.8.0 py36\_0

pytz 2019.2 py\_0

pyyaml 5.1.2 py36h7b6447c\_0

readline 7.0 h7b6447c\_5

scipy 1.3.0 py36he2b7bc3\_0

setuptools 41.0.1 py36\_0

six 1.12.0 py36\_0

sqlite 3.29.0 h7b6447c\_0

tensorflow 1.2.1 py36\_0

tk 8.6.8 hbc83047\_0

tornado 6.0.3 py36h7b6447c\_0

werkzeug 0.15.5 py\_0

wheel 0.33.4 py36\_0

xz 5.2.4 h14c3975\_4

yaml 0.1.7 h1bed415\_2

zlib 1.2.11 h7b6447c\_3

(autokin) [michalos@enki05:~/autokin] $

And the bad news is getting access to the HPC as it is often hogged by a couple of heavy duty users:

[michalos@enki:~/autokin] $squeue

JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)

99297 debug bash michalos PD 0:00 1 (PartitionTimeLimit)

99263 gpu job\_scri snh15 R 5:19:20 1 enki07

99242 gpu vgg\_pola lyg R 16:27:57 1 enki04

99241 gpu vgg\_pola lyg R 16:28:32 1 enki03

99240 gpu vgg\_pola lyg R 16:28:41 1 enki03

99239 gpu vgg\_pola lyg R 16:28:56 1 enki02

99238 gpu vgg\_pola lyg R 16:29:08 1 enki02

99049 gpu vgg\_pola lyg R 4-20:50:36 1 enki06

99061 gpu C2N\_stra smolyan R 3-10:36:37 1 enki01

99251 gpu slurmGpu dja3 R 5:47:28 1 enki13

99121 gpu C2N\_stra smolyan R 2-00:20:48 1 enki12

99116 gpu bash znw7 R 2-00:32:13 1 enki03

99110 gpu bash znw7 R 2-00:48:08 1 enki06

99109 gpu bash znw7 R 2-00:51:05 1 enki04

99108 gpu bash znw7 R 2-00:52:23 1 enki04

99107 gpu bash znw7 R 2-00:53:26 1 enki04

99106 gpu bash znw7 R 2-00:54:53 1 enki02

99074 gpu bash znw7 R 2-04:50:58 1 enki05

99031 gpu bash znw7 R 5-00:37:48 2 enki[02-03]

98924 gpu bash znw7 R 6-22:44:18 2 enki[10-11]

## What is Distributed Deep Learning?

Distributed Deep Learning (DDL) is a MPI-based communication library, which is specifically optimized for deep learning training. An application integrated with DDL becomes an MPI-application, which will allow the use of the ddlrun command to invoke the job in parallel across a cluster of systems. DDL understands multi-tier network environment and uses different libraries (e.g. NCCL) and algorithms to get the best performance in multi-node, multi-GPU environments.

Check out this performance proof-point that shows how DDL maximized research productivity by training on more images at the same time with TensorFlow 1.4.0 running on a cluster of IBM Power System AC922 servers with Nvidia Tesla V100 GPUs connected via NVLink 2.0: [Distributed Deep Learning: IBM POWER9™ with Nvidia Tesla V100 results in 2.3X more data processed on TensorFlow versus tested x86 systems.](https://developer.ibm.com/linuxonpower/perfcol/perfcol-mldl#tab_tensorflow4)

## Anaconda Python Package Environment

(autokin) [michalos@enki13:~/autokin] $conda list

# packages in environment at /home/michalos/envs/autokin:

#

# Name Version Build Channel

\_libgcc\_mutex 0.1 main

\_tflow\_select 2.1.0 gpu\_723.g2e92f34 https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

absl-py 0.7.1 py36\_0

astor 0.7.1 py36\_0

backports 1.0 py\_2

backports.weakref 1.0rc1 py36\_0

blas 1.0 openblas

bleach 1.5.0 py36\_0

c-ares 1.15.0 h7b6447c\_1001

ca-certificates 2019.5.15 1

certifi 2019.6.16 py36\_1

cudatoolkit 10.1.168 533.g8d035fd https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

cudnn 7.5.1\_10.1 507.gcdf2330 https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

cycler 0.10.0 py36\_0

freetype 2.9.1 h8a8886c\_0

gast 0.2.2 py36\_0

google-pasta 0.1.6 py36\_482.gde1dc70 https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

graphsurgeon 0.4.1 py36\_515.gdfe9810 https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

grpcio 1.16.1 py36hf8bcb03\_1

h5py 2.8.0 py36h8d01980\_0

hdf5 1.10.2 hba1933b\_1

html5lib 0.9999999 py36\_0

icu 58.2 h64fc554\_1

keras 2.2.2 pypi\_0 pypi

keras-applications 1.0.4 pypi\_0 pypi

keras-base 2.2.4 py36\_536.gd8f7d7c https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

keras-preprocessing 1.1.0 pypi\_0 pypi

kiwisolver 1.1.0 py36he6710b0\_0

libedit 3.1.20181209 hc058e9b\_0

libffi 3.2.1 hf62a594\_5

libgcc-ng 8.2.0 h822a55f\_1

libgfortran-ng 7.3.0 h822a55f\_1

libopenblas 0.2.20 h9ac9557\_7

libpng 1.6.37 hbc83047\_0

libprotobuf 3.7.1 494.g6f829d7 https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

libstdcxx-ng 8.2.0 h822a55f\_1

markdown 3.1.1 py36\_0

matplotlib 3.1.0 py36h5429711\_0

nccl 2.4.7 501.g7a305d1 https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

ncurses 6.1 he6710b0\_1

numpy 1.14.5 py36h2aefc1b\_4

numpy-base 1.14.5 py36h2b20989\_4

openssl 1.1.1c h7b6447c\_1

pandas 0.25.0 py36he6710b0\_0

pip 19.2.2 py36\_0

powerai-license 1.6.1 627.gc80b92e https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

powerai-release 1.6.1 488.g24b6af4 https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

protobuf 3.7.1 py36\_504.g85ad89d https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

pyparsing 2.4.2 py\_0

python 3.6.9 h2bede3c\_0

python-dateutil 2.8.0 py36\_0

pytz 2019.2 py\_0

pyyaml 5.1.2 py36h7b6447c\_0

readline 7.0 h7b6447c\_5

scikit-learn 0.19.1 py36h6cfcb94\_0

scipy 1.1.0 py36h9c1e066\_0

setuptools 41.0.1 py36\_0

six 1.12.0 py36\_0

sqlite 3.29.0 h7b6447c\_0

tensorboard 1.14.0 py36\_d2a1b9e\_3334.gb94cdc6 https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

tensorflow 1.14.0 gpu\_py36\_733.gb94cdc6 https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

tensorflow-base 1.14.0 gpu\_py36\_87989f6\_57051.gd288838 https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

tensorflow-estimator 1.14.0 py36\_a031b45\_1136.gb94cdc6 https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

tensorflow-gpu 1.14.0 733.gb94cdc6 https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

tensorrt 5.1.3.6 py36\_515.gdfe9810 https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

termcolor 1.1.0 py36\_1

tk 8.6.8 hbc83047\_0

tornado 6.0.3 py36h7b6447c\_0

uff 0.6.3 py36\_515.gdfe9810 https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda

werkzeug 0.14.1 py36\_0

wheel 0.33.4 py36\_0

wrapt 1.10.11 py36h14c3975\_2

xz 5.2.4 h14c3975\_4

yaml 0.1.7 h1bed415\_2

zlib 1.2.11 h7b6447c\_3

(autokin) [michalos@enki13:~/autokin] $