What data analytics tools does NAS provide for machine learning?

For a quick glance at the software that is currently provided in our machine learning environments, see Machine Learning: Overview.

You can also complete these steps to see the installed environments:

1. Load the miniconda3 module:

   ```
   % module use -a /swbuild/analytix/tools/modulefiles
   % module load miniconda3/v4
   ```

2. Run the following command to see the available environments:

   ```
   % conda info --envs
   ```

Is Jupyter Notebook available?

Jupyter Notebook, an open-source application based on Interactive Python, is a useful tool for interactively exploring science data. You cannot run Jupyter on a front-end system (such as a PFE); instead, you must run it on a compute note dedicated to you alone.

To learn more about using Jupyter Notebook on NAS systems, see Using Jupyter Notebook for Machine Learning Development.

Note: A one-time security setup is required before you can begin using Jupyter Notebook.

Which environments can I use to complete Step 2 in the Secure Setup for Jupyter Notebooks?

All of our provided environments have Jupyter installed, so you can choose any of them, besides the base environment. You can find the available environments by using the `conda info -envs` command. We suggest using the latest TensorFlow environment (currently, `tf2_4`).

Do I need to install Jupyter myself?

You do not need to install Jupyter for yourself in order to do thesecure setup step, or use it afterwards. Loading any provided environment will work in the secure setup step, and the completed procedure will allow you to use any of the other environments with Juypter installed.

Can I run a Jupyter Notebook without using the web-based user interface?

You cannot directly run a Jupyter Notebook without using the web-based user interface. However, you can convert your notebook into a Python script and run the script though the regular Python shell command.

To convert a notebook into a Python script:

```
% jupyter nbconvert --to script notebook.ipynb
```

This will create a *.py* file of the specified notebook, with each cell run in sequential order.

How do I run multiple instances of Python on the compute nodes?

To learn how to run multiple instances of a Python code on the compute nodes, with each instance running your code, seeUsing GNU Parallel to Package Multiple Jobs (Instances) in a Single PBS Job.

How do I use multiple nodes with TensorFlow?

To use multiple nodes, you can use TensorFlow's "MultiWorkerMirroredStrategy" variable, which is described in their article Distributed Training with TensorFlow.

To call the strategy in your code, use:

```
if dist_nodes :
    strategy=tf.distribute.experimental.MultiWorkerMirroredStrategy (
          tf.distribute.experimental.CollectiveCommunication.AUTO )
  else :
    strategy = tf.distribute.MirroredStrategy()
```

The collective operations include the following options:

CollectiveCommunication.RING
: Implements ring-based collectives using gRPC as the communication layer.

CollectiveCommunication.NCCL
: Uses NVIDIA's NCCL to implement collectives.

CollectiveCommunication.AUTO
: Defers the choice to the runtime.

You also need to set up the TF_CONFIG environment variable, which is required by TensorFlow:

```
if dist_nodes :
   index = int(sys.argv[1])  # $@ has n indices indicating n nodes
   verbose = index < 1       # verbose is True only when index = 0
   nodes=[]
   # node names for each node, append a port # to the names
   for i in range(2,len(sys.argv)):
     nodes.append(sys.argv[i] + ':2001')
   # set TF_CONFIG variable that MultiWorkerMirroredStrategy needs
   os.environ['TF_CONFIG'] = json.dumps({
      'cluster': {
         'worker': nodes
      },
      'task': {'type': 'worker', 'index': index}
   })
   print(os.environ['TF_CONFIG'])
 ##
```

For more details about TensorFlow programming and an example of how to use the above methodology, please see Multiple CPU Nodes and Training in TensorFlow.