Jupyter at SLAC

Jupyter is a web based analysis and coding environment. It supports multiple different programming languages, but is mostly centered around python development. The main advantage over standard IDEs is that it provide immediate code execution and inline graphics - ie you can interactively explore data.

SLAC implements JupyterHub at http://jupyter.slac.stanford.edu. This provides a central point of access to your jupyter environment using SLAC credentials and access to data stored on SLAC GPFS.

What is somewhat unique to SLAC's implementation of jupyter is that we wish to:

- provide an environment for different experiments to utilise our infrastructure with minimal hassle.
- provide a means for users to run their own jupyter environments (their analysis environment, with their down dependencies)

The current implementation utilises kubernetes and docker images to provide the above functionality.

Link to tutorials

Jupyter on Batch

In order to provide a consistent environment across the web-based jupyter and batch systems, our currently recommendation is to develop your jupyter environment in Docker and then convert the images to Singularity. From there, we can integrate into our modulefiles system directly in order to provide command line access (and hence batch).

```
# modulefiles relies on an ENV MODULEPATH
$ export MODULEPATH=/usr/share/Modules/modulefiles:/etc/modulefiles:/afs/slac/package/singularity/modulefiles:
/opt/modulefiles

# list available modulesfiles
$ module avail
------------------------------------------ /afs/slac/package/singularity/modulefiles
------------------------------------------
...
cdms-jupyterlab/1.6
slac-ml/20181002.0
slac-ml/20190606.1
...
```

There is a specific module called `slac-ml` that provides a prebaked Singularity image derived from the jupyterhub image.

When we load a module, it will override certain environments so that we can now use the 'application' defined in the modulefile:

```
# first we have nothing loaded
$ which jupyter
/usr/bin/which: no jupyter in (/afs/slac/g/scs/net/bin:/afs/slac/u/sf/ytl/sys/bin:/afs/slac/u/sf/ytl/sys//bin:
/usr/afsws/bin:/usr/local/bin:/afs/slac/package/lsf/curr/amd64_rhel70/bin:/opt/hpc/gcc-4.8.5/openmpi-3.1.2/fftw-
3.3.8/bin:/opt/hpc/gcc-4.8.5/openmpi-3.1.2/parallel-hdf5-1.10.4/bin:/opt/hpc/gcc-4.8.5/openmpi-3.1.2/bin:/usr
/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin)

# then we load the module
$ module load slac-ml/20190606.1


# now we see the jupyter exe in our path
$ which jupyter
/afs/slac/package/singularity/images/slac-ml/20190606.1/bin/jupyter
```

In order to run on batch, we will use this modulefile system and submit jobs by creating a text file (eg `myscript.sh`) like

```
#!/bin/bash -l

#BSUB -P jupyter
#BSUB -J my_batch_job_name
#BSUB -q slacgpu
#BSUB -n 1
#BSUB -R "span[hosts=1]"
#BSUB -W 72:00
#BSUB -B

# setup env
source /etc/profile.d/modules.sh
export MODULEPATH=/usr/share/Modules/modulefiles:/opt/modulefiles:/afs/slac/package/singularity/modulefiles
module purge
module load PrgEnv-gcc
module slac-ml/20190606.1

# run the notebook, executing all cells
cd MY_DATA_DIRECTORY
jupyter nbconvert --to notebook --inplace --execute mynotebook.ipynb
```

You can then submit the job to batch via

```
bsub < myscript.sh
```

and you can monitor the job with

```
bjob -l {jobid}
```

We recommend either using nbconvert or papermill to provide parameter access to notebooks.