Installing Your Own Python Packages

Sometimes it is useful to install python packages that are not part of psana-python or install particular versions of a software package. These should be done from the "**psbuild-rhel7**" machines since they have internet access. In particular "psana" nodes do **NOT** have internet access.

Anaconda

You can create your own conda environments with your own choice of packages. You don't need to install your own conda software for this, you can reuse the existing LCLS conda software and use the "conda create" command that you can read about on the web to create your own environment.

One disadvantage of creating your own environments is that they can use significant disk space. See the "Installation Area" section below for a special area that has been created to store conda envs.

If you want to create a new environment with a selected set of packages use a command like this (after sourcing the appropriate psana1/psana2 setup script).

```
conda env create --name roottest root psana
```

If you want to get a conda environment with psana you can clone the psana environment into a local environment that you can control with a command like this (which will take some time because it has to copy several GB to the new environment):

```
conda env create --name my-ana-1.3.10 --clone ana-1.3.10
```

You can also create new (empty) environments without cloning: this is often easier since the psana package is big with quite a few dependencies. You can see a list of available psana environments (to clone) like this:

```
conda info --envs
```

Then activate your new environment with:

```
conda activate my-ana-1.3.10
```

Then install the conda-python package you want (potentially a specific version) using a command like:

```
conda install scipy=0.15.0
```

Note that package version conflicts can make it impossible to install a specific version. One possible way to address this is to remove the package requiring the conflicting version using:

```
conda remove <packagename>
```

Installation Area

A special installation area has been created for users to install conda environments to avoid using up home-directory quota space. Use commands similar to these to create your own environments.

```
source /sdf/group/lcls/ds/ana/sw/conda1/manage/bin/psconda.sh  # for LCLS1 environments. replace "conda1" with
"conda2" for LCLS2
# for the following 3 commands, change "my-h5-1.0.0" to the name of your environment
export CONDA_PKGS_DIRS=/sdf/group/lcls/ds/tools/conda_envs/my-h5-1.0.0/.pkgs # avoids storing large package
files in your home directory
conda create --prefix /sdf/group/lcls/ds/tools/conda_envs/my-h5-1.0.0 --clone h5-1.0.0 # or "create" command
from the previous section
chmod -R +r /sdf/group/lcls/ds/tools/conda_envs/my-h5-1.0.0 # make directory world-readable
```

Anaconda+pip

Some packages are not available from anaconda but they can be installed via pip (in this case the "rayopt" package):

```
source activate ana-1.3.44
pip install --user rayopt
```

If you change psana versions at a later time, the package you installed previously with pip may not work anymore. If so you can uninstall the old version with "pip uninstall rayopt" and install it again using the new version of psana.

Making Your Own Conda Environment Visible In Jupyter

If you build your own conda environment - or want to use one of your colleagues one - which contains a python jupyter kernel (package *ipykernel*) you can make it visible to the jupyter hub by creating a kernelspec files. This is also useful if you want to have a kernel that points to an old psana release. To do that you can copy/modify kernel files from /reg/g/psdm/sw/conda/jhub_config/prod-rhel7/kernels/ in the subdirectories. The subdirectories ana1-current, ana1-current-py3 are for LCLS1 psana (py2 and py3 versions) and subdirectory ana2-current for LCLS2 psana (py3 only).

For a user the spec files are located in ~/.local/share/jupyter/kernels/ with a sub-directory for each spec. The *kernel.json* in a spec directory contains the settings. For an example see ~wilko/.local/share/jupyter/kernels/example/.

An easy way to create the proper folder and spec files is to use the ipython command:

```
% conda activate <your-env>
% ipython kernel install --user --name "a-name-you-like"
```

You can also copy the files

The --name option is the name for the kernelspec - the directory that will be created in ~/.local/share/jupyter/kernels/. There are a few more install options - *i python kernel install -h* - the most useful one is *--display-name* that allows to set the name shown in Jupyter. Instead of using the command above one can also create the directory and the *kernel.json* file by hand. You

Virtualenv

NOTE: the virtualenv method is not compatible with our anaconda-based psana (so not recommended as the first approach) but we leave the documentation here in case it is useful at some point.

A python virtualenv can be used to install python packages locally in your user account. Some instructions are here. Execute this command to create a new virtual environment (after setting up the psana environment):

```
virtualenv <directoryName>
```

where you replace <directoryName> with the name of the directory you would like to create. Then activate your virtualenv like this:

```
source <directoryName>/bin/activate.csh     #use this for tcsh

or

source <directoryName>/bin/activate         #use this for bash
```

Use "deactivate" to exit the virtualenv. Install various packages using "pip install <packageName>" as shown in these instructions. Virtualenv packages can be used in batch jobs, but you must submit the job while the environment is active.