Jupyter Notebooks

Adding additional packages to Jupyter Notebooks

Google the desired package (python, R packages, not OS packages) and review the installation instructions. They might recommend a different conda repository than the default.
Be careful if conda says it wants to downgrade packages. If it is a minor downgrade, it is probably OK. Do not proceed if many packages must be downgraded or critical packages are to be downgraded. Remember that changes to the Ananconda environment will affect all tools using the environment and a downgrade could cause tools not to function (if a feature is no longer available, for example).

- Start a workspace tool. If your HUB supports multiple operating system (vendor or version) based tool containers choose the workspace tool for the appropriate container.
- As an alternative, start the Jupyter Notebook tool based on the Anaconda environment that you want to modify and start a new terminal.
- From the open terminal switch to the apps user (your account must be a member of the apps group).

    - `sudo su - apps`

- Load the Anaconda environment that you wish to modify (there may be multiple Anaconda environments available. The list of Anaconda environments can be listed with the command

    - `use |& grep anaconda`

- Execute the command "use", choosing the appropriate Anaconda (X represents the particular version choice) environment.

    - `use -e -r anaconda-X`

- Install the desired package via conda or mamba if it is installed. mamba is typically much more efficient and provides more useful information should it fail.   Another option is to use pip. Extra care must be taken when mixing conda and pip package installations in the same environment. Typical installation command like

    - `conda install -c conda-forge -c defaults <pkgname>`
    - `pip install -U --upgrade-strategy only-if-needed <pkgname>`

The installation may take a few minutes

- **Important! Fix any world writable files by doing:**
    - `chmod -R o-w /apps/share64/<OS>/anaconda/anaconda-X`

Creating a separate anaconda environment.

In some circumstances it is beneficial to spawn a new named environment from the standard base environment. The named environments are referred to as kernels in Jupyter notebooks. Once a Jupyter notebook is running the developer or user has the opportunity to change to a different kernel. The kernel used to run the notebook is embedded within the notebook metadata therefore only has to be changed once. This works well for developers and means the administrator does not have to create a Jupyter Notebook tool for each kernel.

- Set base Anaconda environment
    - `use -e -r anaconda-X`
- Create new named environment
    - `conda create -n <name>`
- Activate named environment
    - `source activate <name>`
- Do package installation as before
    - `conda install -c conda-forge -c defaults <pkgname>`
    - `pip install -U --upgrade-strategy only-if-needed <pkgname>`
- Register named environment as kernel in base environment
    - `python -m ipykernel install --sys-prefix --name <name> --display-name "Python3 (<name>)"`
    - `python -m ipykernel install --prefix /apps/share64/<OS>/anaconda/anaconda-X --name <name> --display-`

```
name "Python3 (<name>)"
```

- Deactivate named environment
    - `conda deactivate`

References:

conda user guide

Ipython kernel instalation

Updating hublib

The HUBzero utility library should be installed in every base Anaconda environment and in each named environment as well. The hublib package is available only through pip.

- Set Anaconda base environment
    - `use -e -r anaconda-X`
- Install hublib
    - `pip install -U hublib`