Jupyter on O2

Due to implementation choices by the Jupyter developers, Jupyter and its associated dependencies are not installed by default on O2. However, we acknowledge that many users leverage Jupyter notebooks to great effect in their research, so we provide detailed instructions on how to set up a working Jupyter notebook here on O2.

Please note: We provide instructions for setting up Jupyter on O2, though we are only available for limited support of this use case. We are investigating the feasibility of offering a more robust solution in the future.

Installing Jupyter

As mentioned above, Jupyter is not installed into any of the Python installations available via the LMOD module system. However, it is very straightforward to install locally, via the use of a virtual environment. For detailed instructions on setting up a virtual environment, see Personal Python Packages, but the instructions will be reiterated here specifically for installing Jupyter.

First, create your virtual environment. We use version 2.7.12 to demonstrate in this example. Additionally, we install the virtual environment to our hypothetical home directory in this example, but you may create and use virtual environments wherever is convenient.

```
ECOMMONS@login01:~$ module load gcc/6.2.0 python/2.7.12
ECOMMONS@login01:~$ virtualenv jupytervenv --system-site-packages
```

Recall from Using Applications on O2 and exploration via `module avail` that the `python/2.7.12` module will not be visible nor loadable until `gcc/6.2.0` is loaded.

Now, source the environment, and install jupyter:

```
ECOMMONS@login01:~$ source jupytervenv/bin/activate
(jupytervenv)ECOMMONS@login01:~$ pip install jupyter
```

At this point, a number of packages will attempt to install, but hopefully will not throw any errors.

Opening a Notebook

The above steps only need to be taken once (unless you need to recreate the virtual environment, or build another one). After the installation finishes, open a new LOCAL terminal. The following instructions assume you will be connecting from OS X or some other native *nix terminal (e.g. via Debian/CentOS, etc.). If on Windows, use some sort of terminal emulator such as Cygwin or MobaXterm (further configuration may be required, e.g. for X11 forwarding).

Make sure you have X11 forwarding active (e.g. XQuartz is running if on a Mac). Pick a port on your local machine that is empty using whatever methods you like (e.g. `netstat`). Generally, somewhere in the 50000 range is safe if you just want to guess. SSH to O2 with that port (to be mentioned as PORT):

```
me@localhost:~$ ssh -Y -L PORT:127.0.0.1:PORT ECOMMONS@o2.hms.harvard.edu
```

Remember which login host you land on. For example, let's pretend we landed on login01. Now, request an interactive session:

```
ECOMMONS@login01:~$ srun -t 0-3:00 --pty -p interactive --x11 /bin/bash
```

If you might require multiple cores and more than 1GB of memory for your notebook, also specify that here, via `-n` and `--mem=`. See Using Slurm Basic or the sbatch man page for more sbatch flags. Here, `--pty` and `--x11` are mandatory.

Remember which compute node you landed on. Let's pretend we landed on compute-a-16-20. Start the virtual environment and notebook (Jupyter and your virtual environment will require compiler libraries, so you'll need to load GCC as well):

```
ECOMMONS@compute-a-16-20:~$ module load gcc/6.2.0 python/2.7.12
ECOMMONS@compute-a-16-20:~$ source jupytervenv/bin/activate
(jupytervenv)ECOMMONS@compute-a-16-20:~$ jupyter notebook --port=PORT
--browser='none'
```

Alternately, to open an existing notebook:

```
(jupytervenv)ECOMMONS@compute-a-16-20:~$ jupyter notebook NOTEBOOKFILE
--port=PORT --browser='none'
```

On newer versions of Jupyter (notebook >= 4.1), the developers implemented token authentication, which is on by default. If you have a freshly installed version of Jupyter, your notebook will have a token associated with it. When you run the above command, you should see a URL show up in the terminal that contains your session token. **Remember this URL AND KEEP THIS SESSION ACTIVE.**

Now, open a new, **SECOND** local terminal. First, SSH to the login node you landed on at the beginning of the process (e.g. login01):

```
me@localhost:~$ ssh ECOMMONS@login01.o2.rc.hms.harvard.edu
```

Then, SSH directly to the compute node where the notebook is open (e.g. compute-a-16-20), using the same port as before:

```
ECOMMONS@login01:~$ ssh -N -L PORT:127.0.0.1:PORT compute-a-16-20
```

Your session will appear to hang here. This is expected. Now, if you are using a newer version of Jupyter, go back to the previous terminal window, and **copy the URL that the notebook generated, and paste it into your local browser**. You should now be able to view your notebook! If you are using an older version of Jupyter, or have disabled token authentication, you can manually visit **127.0.0.1:PORT** in your local browser to access the notebook. If you have done everything correctly, you should have two terminal sessions active: one displaying the log for the active notebook, one hanging session with no visual feedback, and your open browser tab with the notebook interface.

Using other programming languages / Jupyter kernels

The instructions above will enable you to create a Juypter notebook that supports running Python. There are a number of other programming languages that can be used with Jupyter, which simply require installing the appropriate kernel to include support for a given programming language. (Python support is automatically included when the jupyter package is installed to your virtual environment, as the IPython kernel is the default kernel for Jupyter).

If you want to use a Jupyter-supported programming language other than Python, you will need to manually install the appropriate kernel.

Please note that we have only tested the use case of non-standard kernels with IRkernel, which allows you to run R notebooks using Jupyter. Using any other kernels for support of programming languages other than Python or R through Jupyter on O2 may be done without any implied support or guarantee of functionality.

R kernel for Jupyter

For example, if you want to run R through a Jupyter notebook on O2, you need to install the IRkernel package to a personal R library. This should be done after the Installing Jupyter instructions, and prior to the Opening a Notebook instructions. First, get into an interactive session:

```
ECOMMONS@login01:~$ srun --pty -p interactive -t 0-2 bash
```

While in an interactive session, set up the personal R library. Here we're using the R-3.4.1 module, so the R library reflects this version number in its name:

```
ECOMMONS@compute-a-16-68:~$ mkdir -p ~/R-3.4.1-IRkernel/library
ECOMMONS@compute-a-16-68:~$ echo 'R_LIBS_USER="~/R-3.4.1-IRkernel/library"'
> $HOME/.Renviron
ECOMMONS@compute-a-16-68:~$ export R_LIBS_USER="~/R-3.4.1-IRkernel/library"
```

Next, install IRkernel and dependencies while your virtual environment is sourced:

```
ECOMMONS@compute-a-16-68:~$ module load gcc/6.2.0 python/2.7.12 R/3.4.1
ECOMMONS@compute-a-16-68:~$ source jupytervenv/bin/activate
(jupytervenv) ECOMMONS@compute-a-16-68:~$ R
> install.packages(c('repr', 'IRdisplay', 'evaluate', 'crayon', 'pbdZMQ',
'devtools', 'uuid', 'digest'))
# select mirror
> devtools::install_github('IRkernel/IRkernel')
> IRkernel::installspec()
```

Installing the IRkernel and dependent R packages to a personal R library will only need to be done once. At this point, you can proceed to the Opening a Notebook instructions. When your notebook is opened, you should now see an option for creating an R notebook under the "New" button.

In the future, you'll should ensure that your `~/.Renviron` file and `R_LIBS_USER` environment variable point to the correct R personal library before trying to open an R notebook in Jupyter.

Opening an Rshiny application with Jupyter

You may encounter an R package that deploys an Rshiny app that you can access; on your desktop, this is as easy as pasting the provided link into the browser. On O2, an extra step needs to be taken.

When the URL is generated, it should specify a port number at the end. You need to repeat the last step of the above port forwarding procedure for this new port. For example, say your RShiny URL looks something like `127.0.0.1:7875` on `compute-a-16-20`. You'll need to open a **THIRD** terminal session (in addition to the two you already have open after following the above two sessions), and open a tunnel using this specified port:

```
me@localhost:~$ ssh -Y -L 7875:127.0.0.1:7875 ECOMMONS@o2.hms.harvard.edu
...
ECOMMONS@login01:~$ ssh -N -L 7875:127.0.0.1:7875 compute-a-16-20
```

And this session will hang just like the second session above. So, you should now have **THREE** open sessions: one with the notebook log, one hanging session that is keeping the notebook tunnel open, and one hanging session that is keeping the RShiny tunnel open. Once you have confirmed this, you can now paste the RShiny app URL into your local browser and navigate the app accordingly.

Note that each time you generate the RShiny app URL, it is possible that the port will change each time (e.g. across sessions). This means you need to re-execute the third session every time (don't forget to close the previous session).