Upgrading THREDDS and Deploying JupyterHub at the University of Wisconsin-Milwaukee to Support Education and Research

Final Report, Clark Evans (evans36@uwm.edu), 20 April 2023

Thanks to generous financial support provided by Unidata in summer 2021, the University of Wisconsin-Milwaukee Atmospheric Science Program purchased one Dell PowerEdge T640 server to enable us to continue our IDD participation and deploy a local The Littlest JupyterHub instance. Since its deployment in August 2021, the server, data we receive through the IDD, and the local JupyterHub instance have supported student training and development in four Atmospheric Science courses and through our Innovative Weather student traineeship program.

Our undergraduate Atmospheric Science major transitioned its programming requirement from FORTRAN to Python effective in Fall 2021. This allowed us to modernize our two-semester Synoptic Meteorology I and II lab curriculum, wherein students gain experience in generating meteorological analyses using Python rather than using instructor- and/or web-generated analyses, and to develop a new lab-based course, Introduction to Climate Science, wherein students use Python resources to interrogate data and deepen their understanding of the Earth's climate. Concurrently, we integrated Python into our graduate-level Numerical Weather Prediction course, wherein students use Python resources to visualize and analyze outputs from toy models and the full-physics WRF-ARW model. Our The Littlest JupyterHub instance was used to deploy Jupyter Notebooks and provide a web-based Python programming and visualization environment in support of these courses beginning in Fall 2021 (Numerical Weather Prediction) and continuing to the present (Spring 2023; Introduction to Climate Science and Synoptic Meteorology II). Students and instructors both appreciate having a consistent web-based environment they can access from anywhere, using nearly any device, to complete course assignments and activities, and our Unidata-funded resources will continue to be a part of these courses for years to come. Altogether, over twenty students have been introduced to Unidata and related technologies through these courses.

Jupyter Notebooks developed in support of the Synoptic Meteorology I lab are available at:

https://github.com/evans36/SynopticMet

The Jupyter Notebooks developed in support of the Synoptic Meteorology II lab will be available from the same repository shortly after the course's completion in May 2023.

Jupyter Notebooks developed in support of Numerical Weather Prediction are available on GitHub in two repositories, one containing example notebooks for working with WRF-ARW model data (as is a part of three course assignments and the term project) and one containing example notebooks for working with text-based and GRIB-formatted meteorological data (as is a part of two course assignments):

https://github.com/evans36/wrf-python-notebooks
https://github.com/evans36/data-wrangling-examples

The Introduction to Climate Science course leverages Jupyter Notebooks provided by Prof. Eli Tziperman at Harvard University as part of his 2022 textbook, *Global Warming Science: A Quantitative Introduction to Climate Change and its Consequences*, and are available at:

https://courses.seas.harvard.edu/climate/eli/Courses/EPS101/index.html

To assist other universities who may wish to deploy a local JupyterHub using The Littlest JupyterHub in support of teaching and/or research, we have prepared a short guide to doing so that is included as an appendix to this report. Please feel free to broadly share this guide, and we welcome any additions, clarifications, or questions!

In addition to its use inside our formal curricula, the Unidata-funded server has been used in our Innovative Weather program, which trains students in the rigors of operational meteorology to provide impact-based forecasts and decision support to public- and private-sector clients across the Midwest United States. Data received through the IDD are locally parsed using automated Python scripts into meteorological analyses routinely used by the student forecasters as part of their forecast-preparation process. Likewise, these scripts themselves were developed by student interns using our local The Littlest JupyterHub instance as a pre-deployment sandbox. Altogether, over ten additional students have been introduced to Unidata and related technologies through these courses.

Appendix: Configuring and Deploying The Littlest JupyterHub

Basic Configuration

Ubuntu 20.04 LTS – required for The Littlest JupyterHub; the LTS version was selected over the most-recent distribution (21.04 at the time of writing) to ensure the OS's prolonged longevity. We followed standard procedures to install Ubuntu except for manually partitioning the hard drive. We chose this so that we could create a large, dedicated data drive for the LDM.

Installation and Initial Configuration

We installed the current version as of 9 August 2021 (v0.1); this version runs Python 3.7 (which is a version below the Ubuntu 20.04 LTS's Python 3.8 default). We followed the instructions for installing The Littlest JupyterHub on your own server at:

Installing on your own server — The Littlest JupyterHub v0.1 documentation

For the initial installation, we followed only Steps 1 (Installing The Littlest JupyterHub) and 4 (Setup HTTPS). We saved Steps 2 (Adding more users) and 3 (Install conda/pip packages for all users) for after configuring the GitHub OAuth authenticator (described below).

We recommend creating a dummy admin account with a dummy password to use for initial testing. After you have set up HTTPS, you can create a new admin user with your desired password, from which you can then remove the initial dummy admin account from both admin and JupyterHub access.

When setting up HTTPS, we followed the instructions at:

Enable HTTPS — The Littlest JupyterHub v0.1 documentation

We were able to follow the "Automatic HTTPS with Let's Encrypt" instructions without issue.
This June 2021 Jupyter blog post also provides some helpful information to support the formal The Littlest JupyterHub documentation:

Setting up a "Production Ready" TLJH deployment | by yuvipanda | Jupyter Blog

Authentication

The Littlest JupyterHub's default authentication method is the "First Use Authenticator," which allows anyone to create an account but requires that a local JupyterHub admin approve the new account before it can be used to log in. As an initial step to further securing the JupyterHub, we used the users.allowed authenticator option to limit which usernames could be used to log in to the JupyterHub. To do so, we ran the following commands (as an admin user) from the JupyterHub terminal:

```
sudo tljh-config add-item users.allowed user1
sudo tljh-config reload
```

More information on specifying allowed users is available under “User Lists” at:

Configuring TLJH with tljh-config — The Littlest JupyterHub v0.1 documentation

After initial deployment, we switched to using the GitHub OAuth authenticator, following the instructions at:

Authenticate using GitHub Usernames — The Littlest JupyterHub v0.1 documentation

Our rationale for using GitHub was two-fold. First, having students set up a GitHub account provides them with a place to store and share code, including Jupyter Notebooks, with others. Second, our institution does not subscribe to Google Workspace, such that the process of using Google’s OAuth authenticator is somewhat more convoluted and limited than that depicted in The Littlest JupyterHub’s documentation.

A few notes on using the GitHub authenticator:

- The Littlest JupyterHub assigns GitHub users to a username that matches their GitHub user ID. Having students sign up using a user ID that matches their university student ID or follows a standard convention (e.g., first.last) may help manage JupyterHub access.
- The `tljh-config users.allowed` option works for the usernames associated with GitHub logins. If you are using this option to restrict user access to your JupyterHub, you will need to add the desired username(s) to this access list *before* trying to log in via GitHub authentication.
- *Do not log out of your admin account until you have granted admin access to your GitHub-based account!* Instructions on how to do so are available at Add / Remove admin users — The Littlest JupyterHub v0.1 documentation. We found that adding an admin user from the JupyterHub interface granted that user admin access but did not add them to the tljh-config admin users. Thus, you may wish to follow *both* sets of instructions from the documentation. As noted therein, a new admin user needs to stop and (re-)start their server from the JupyterHub Control Panel to get admin powers if they are active at the time their account is granted admin powers.

More information on The Littlest JupyterHub security considerations, particularly as it relates to user access and user control, is available at:

Security Considerations — The Littlest JupyterHub v0.1 documentation

Adding Packages

The instructions in Step 3 (“Install conda / pip packages for all users”) of the installation guide provide

a short guide to, well, installing packages for all users via conda or pip. More details are provided in the longer guide at:

Install conda, pip or apt packages — The Littlest JupyterHub v0.1 documentation

It's a good idea to update conda to its most-recent version (from that provided with The Littest JupyterHub distribution) before installing any packages. To do so, open a terminal window in JupyterHub and then run:

```
sudo -E conda update -n base -c defaults conda
```

Note the inclusion of sudo -E at the outset, following the guidance in the link above.

After updating conda (and other packages), we decided to first make available a basic set of packages common to many meteorological applications using conda via conda-forge:

```
sudo -E conda install -c conda-forge numpy scipy pandas matplotlib cartopy metpy
siphon shapely fiona netcdf4 xarray dask windspharm cfgrib wrf-python
```

Most of these packages are probably familiar to active Python users. Among those that may not be, siphon is a Unidata package for accessing data from a THREDDS server, shapely and fiona are both shapefile readers, windspharm provides a set of routines for computing kinematic fields such as the non-divergent and irrotational wind, cfgrib is ECMWF's package for reading GRIB files (particularly within xarray), and wrf-python provides routines for working with WRF-ARW model output. Feel free to customize this list for your own installation, though you can always install additional packages later. And, as conda is wont to do, it may take a while to solve the environment and proceed to installing the desired packages.

Establishing Resource Limits

By default, The Littlest JupyterHub does not restrict how much memory or how many CPUs any individual user can use. Even with a powerful machine, it's likely a good idea to establish some limits on memory and CPU usage, recognizing that this can always be changed if necessary.

The Littlest JupyterHub's documentation for the tljh-config package provides documentation on how to establish these limits:

Configuring TLJH with tljh-config — The Littlest JupyterHub v0.1 documentation

We chose to limit individual users to 16 GB of RAM and 4 CPUs:

```
sudo tljh-config set limits.memory 16G
sudo tljh-config set limits.cpu 4
sudo tljh-config reload
```

Terminating Idle Notebook Servers

By default, The Littlest JupyterHub terminates *inactive* notebook servers (i.e., those in which commands are not actively being typed or executed) after ten minutes. An *active* notebook server has no limit on how long it can remain active, however. Instructions on how to change the time after which inactive servers are culled are available at:

Culling idle notebook servers — The Littlest JupyterHub v0.1 documentation

We chose to extend the default termination time to 30 minutes while maintaining no limit on how long an active notebook server can run.

Making Notebooks and Data Available to Users

Since many of our students have never used a JupyterHub or even a Jupyter Notebook before, we wanted to provide a notebook (accessible to them in their JupyterHub home directory on login) that introduced the system and provided links to references for more information. The Littlest JupyterHub provides a convenient process for doing so:

Share data with your users — The Littlest JupyterHub v0.1 documentation

In our case, we followed Option 3 ("Create a directory for users to share Notebooks and other files") to create a place on the server to store this notebook. This procedure results in a symbolic link to the shared directory being created in all *new* users' JupyterHub home directories; *existing* users can manually create a symbolic link to this directory by loading a JupyterHub terminal window, then (locally) running:

```
ln -s /path/to/shared/directory .
```

The resulting link will have the same name as the shared directory itself.

When any user is ready to share a notebook with others, they can copy or move it to the shared directory from the JupyterHub main interface. They will retain edit privileges on this file; all other users will only be able to read the file, however.

It is also possible to make entire GitHub repositories available to users via `nbgitpuller`, which is distributed with The Littlest JupyterHub. For example, we wished to make Jake VanderPlas's *Python Data Science Handbook* (for which Jupyter Notebooks are available in a public GitHub repository) available to students on our JupyterHub with one click. To do so, an admin uses the `nbgitpuller` link generator to indicate what repository (and branch of said repository) to clone and into which JupyterHub to clone the repository. They may also indicate which file (e.g., a particular notebook) should be loaded after the repository has finished being cloned. Users then click on the resulting link to

clone (first use) or update (subsequent uses) the repository in their JupyterHub environment.

Miscellaneous Usage Documentation

- *Tab autocompletion in Jupyter Notebook*: The iPython interpreter in Jupyter Notebook includes its Tab autocompletion extension. However, the default Jedi extension that enables Tab autocompletion can work slowly. Thus, it is recommended to disable the Jedi extension in any Jupyter Notebooks in which you wish to use Tab autocompletion by adding the following line at the top of the notebook:

```
%config Completer.use_jedi = False
```

- Enabling Jupyter Notebook extensions: Enabling Jupyter Notebook extensions — The Littlest JupyterHub v0.1 documentation

- GitHub - plasmabio/tljh-repo2docker: Plugin for The Littlest JupyterHub to build multiple user environments with repo2docker is a TLJH plugin that uses Docker images to facilitate multiple user environments from GitHub-hosted repositories (which contain environment.yml files listing the packages to install). Since this uses Docker, however, the associated environments cannot be updated, added to, etc.

- This bug report lists some configuration steps (e.g., additional kernels, such as for R, and other plugins): `jupyterhub-nativeauthenticator v0.0.7` breaks the sign up page. · Issue #685 · jupyterhub/the-littlest-jupyterhub · GitHub