



# Improving your Python Notebook Workspace with Custom JupyterLab Extensions



Michael Thomas Wells

May 24 - 26



Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

- Why JupyterLab?
- JupyterLab Demonstration
- Customize or Not?
- Custom JupyterLab Extensions
  - Server Extension
  - React Widget
- Documentation
- Deployment
  - conda-pack
- Deployment
  - localhost vs remote



Free software, open standards, and web services for interactive computing across all programming languages

# Why JupyterLab?



- “Individuals and interactions over processes and tools” <http://agilemanifesto.org/>
- Scientists and Researchers need to
  - Share results quickly and in a compelling way
  - Share results with audiences of diverse backgrounds
  - Have direct access to data
  - Adapt to changes in funding
- Enables quick start up
- Flexible to different domains of study

# JupyterLab Demonstration



- <https://jupyter.org/try-jupyter/lab/>

The screenshot displays the JupyterLab web interface in a browser. The address bar shows the URL <https://jupyter.org/try-jupyter/lab/>. The interface includes a top navigation bar with tabs for 'Getting Started', 'Sandbox Proxy - Conf...', 'WSL [Windows Sub...', 'Welcome to Terrafo...', 'COVID-19 Info', 'Search - Nexus Rep...', 'IDEAS', 'WASP/WAVE', 'GMS', 'Health Benefits', 'python', 'Machine Learning 2...', 'Bigfut', 'Digital Presence', 'Cinefin framework', and 'Review and Approval'. Below this is a menu bar with 'File', 'Edit', 'View', 'Run', 'Kernel', 'Tabs', 'Settings', and 'Help'. The left sidebar shows a file explorer with a search bar and a list of notebooks: 'Intro.ipynb' (a day ago), 'Lorenz.ipynb' (21 hours ago, selected), and 'sqlite.ipynb' (21 hours ago). The main area displays the 'Lorenz.ipynb' notebook, which has a title 'The Lorenz Differential Equations'. The notebook content includes a text cell with 'Before we start, we import some preliminary libraries.' followed by a code cell with the following Python code:

```
[ ]: import numpy as np
from matplotlib import pyplot as plt
from scipy import integrate

import piplite
await piplite.install('ipywidgets')

from ipywidgets import interactive, fixed
```

Below the code cell is another text cell: 'We will also define the actual solver and plotting routine.' followed by a code cell with the following Python code:

```
[ ]: def solve_lorenz(sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""
    max_time = 4.0
    N = 30

    fig = plt.figure(1)
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim([-25, 25])
    ax.set_ylim([-35, 35])
    ax.set_zlim([5, 35])

    def lorenz_deriv(x,y,z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x,y,z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random.random((N, 3))

    # Solve for the trajectories
    t = np.linspace(0, max_time, int(250*max_time))
    x_t = np.asarray([integrate.odeint(lorenz_deriv, x0i, t)
                      for x0i in x0])
```

The right sidebar contains a 'Pyolite' section with a message: 'We will also define the actual solver and plotting routine.' Below this are sections for 'Cell Tags' (with an 'Add Tag +' button), 'Slide Type' (a dropdown menu), 'Raw NBConvert Format' (a dropdown menu), and 'Advanced Tools' (a button with a plus sign). The bottom status bar shows 'Simple' mode, '0' kernels, '3' tabs, 'Pyolite | Busy', 'Mode: Command', and 'Ln 1, Col 1 Lorenz.ipynb'.

## Customize or Not?

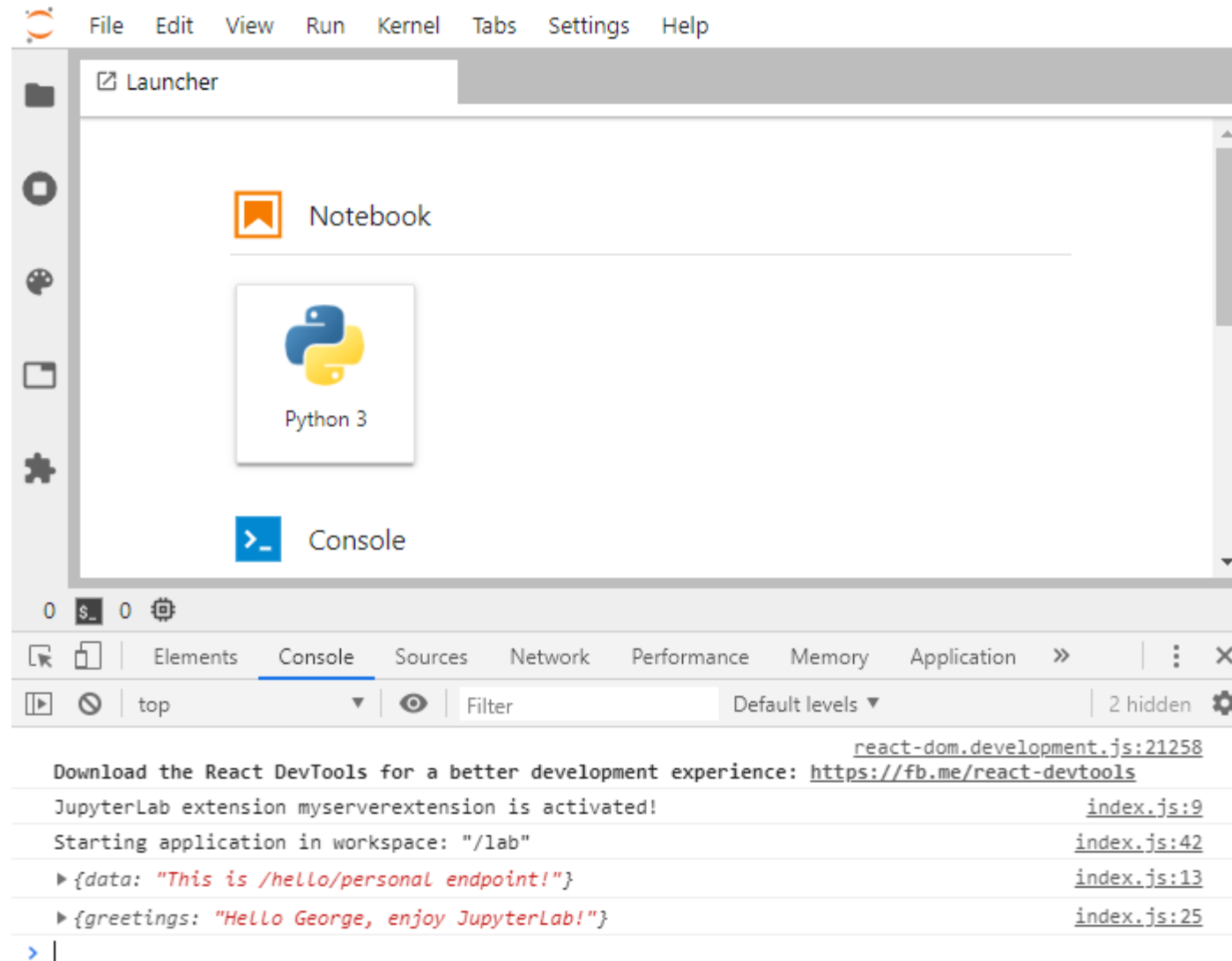


- I have a custom file format that I want to display.
- I want to share my JupyterLab notebook to non technical users.
- I want other researchers to be able to repeat my work with little to no setup.
- I want to minimize the amount of unnecessary implementation details the user sees.

# Custom JupyterLab Extensions



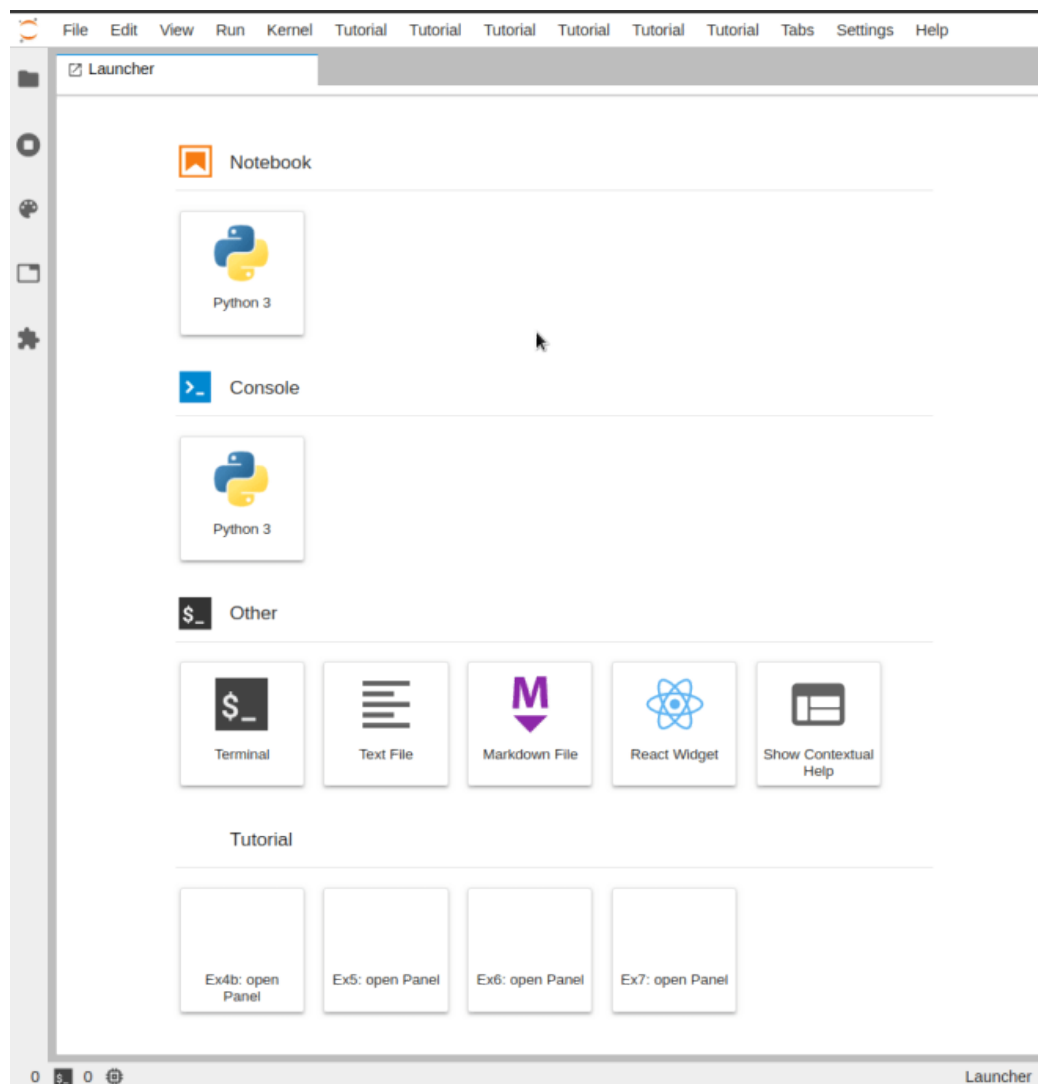
- Server Extension
  - <https://github.com/jupyterlab/extension-examples/tree/master/server-extension>



# Custom JupyterLab Extensions



- React Widget Extension
  - <https://github.com/jupyterlab/extension-examples/tree/master/react-widget>



<https://jupyterlab.readthedocs.io/en/stable/user/extensions.html>

## Extensions

- ⊕ Installing Extensions
- ⊕ Managing Extensions with `jupyter labextension`
- ⊕ Managing Extensions Using the Extension Manager

JupyterLab on JupyterHub

Exporting Notebooks

Localization and language

Real Time Collaboration

Advanced Usage

### EXTENSION DEVELOPER GUIDE

Extension Developer Guide

Common Extension Points

Reusing JupyterLab UI

Documents

Notebook

React

[Docs](#) » Extensions

 Jupyter |  [Edit on GitHub](#)

## Extensions

Fundamentally, JupyterLab is designed as an extensible environment. JupyterLab extensions can customize or enhance any part of JupyterLab. They can provide new themes, file viewers and editors, or renderers for rich outputs in notebooks. Extensions can add items to the menu or command palette, keyboard shortcuts, or settings in the settings system. Extensions can provide an API for other extensions to use and can depend on other extensions. In fact, the whole of JupyterLab itself is simply a collection of extensions that are no more powerful or privileged than any custom extension.

For information about developing extensions, see the [developer documentation](#).

### Table of contents

- [Installing Extensions](#)
- [Managing Extensions with `jupyter labextension`](#)
- [Managing Extensions Using the Extension Manager](#)



# Deployment



- Conda-Pack
  - Sharing JupyterLab extensions
  - <https://conda.github.io/conda-pack/>
  - Demonstration

## conda-pack

A tool for packaging and distributing conda environments.



build passing

## Navigation

[CLI Docs](#)

[API Docs](#)

[Usage with Apache](#)

[Spark on YARN](#)

[Parcels](#)

[SquashFS](#)

## Need help?

Open an issue in the [issue tracker](#).

## Quick search

## Conda-Pack

`conda-pack` is a command line tool for creating archives of [conda environments](#) that can be installed on other systems and locations. This is useful for deploying code in a consistent environment—potentially where python and/or conda isn't already installed.

A tool like `conda-pack` is necessary because conda environments *are not relocatable*. Simply moving an environment to a different directory can render it partially or completely inoperable. `conda-pack` addresses this challenge by building archives from original conda package sources and reproducing conda's own relocation logic.

```
jcrist computer_one $ source activate example
(example) jcrist computer_one $ # Package the current environment
(example) jcrist computer_one $ conda-pack
Collecting packages...
Packing environment at '/Users/jcrist/anaconda/envs/example' to 'example.tar.gz'
[##                               ] | 6% Completed | 0.7s
```



# Deployment

- Localhost
  - Demonstration
- Remote
  - Demonstration



Linux



Mac

