

Introduction to Elyra: AI-centric extensions to JupyterLab

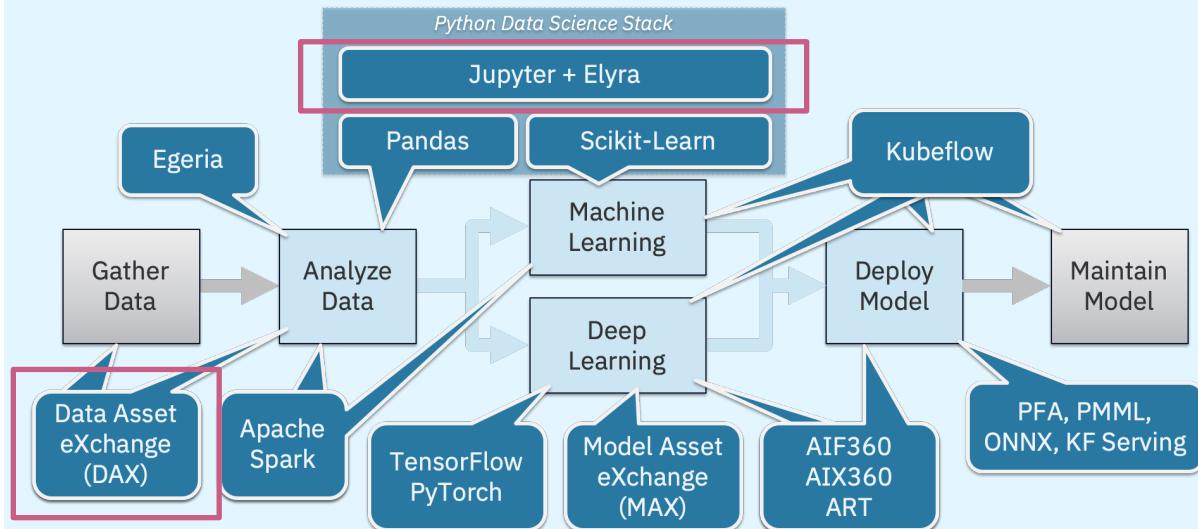
—
Yiwen Li

Developer Advocate
IBM CODAIT

yiwen.Li@ibm.com

- CODAIT aims to make AI solutions dramatically easier to create, deploy, and manage in the enterprise.
- We contribute to and advocate for the open-source technologies that are foundational to IBM's AI offerings.
- 30+ open-source developers!

Improving the Enterprise AI Lifecycle in Open Source





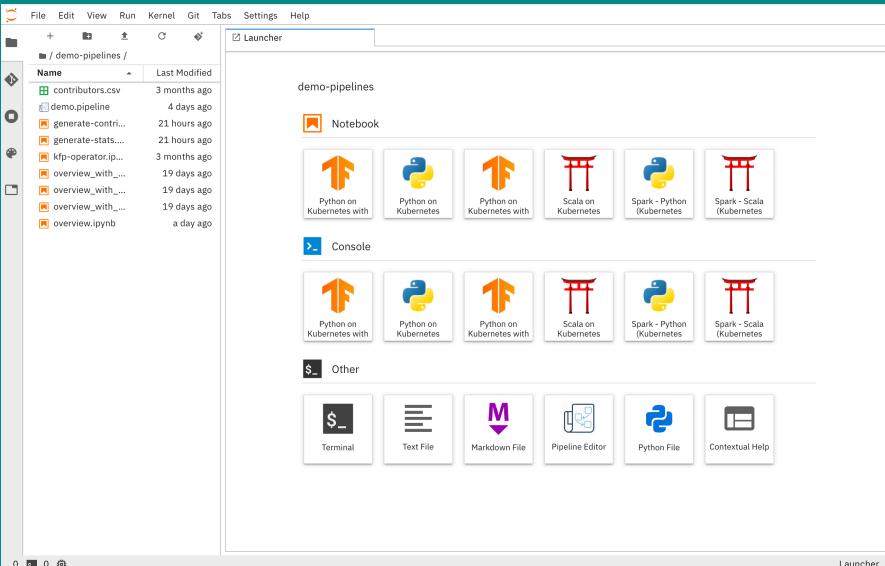
Elyra is a set of AI centric extensions to JupyterLab. It aims to help data scientists, machine learning engineers and AI developer's through the model development life cycle complexities.

Elyra at GitHub

<https://github.com/elyra-ai/elyra>

Elyra Documentation

<https://elyra.readthedocs.io/en/latest/>





Elyra is a set of
AI centric extensions for
JupyterLab

Elyra was officially announced as
an open source project by IBM on
April 29th.

The name Elyra is a word play with
one of the Jupyter moons “Elara”
where we introduce the “y” from
“Jupyter” to make it “Elyra”

The screenshot shows the JupyterLab interface with the Elyra extension installed. On the left, there is a file browser showing a directory named 'demo-pipelines' containing several files: contributors.csv, demo.pipeline, generate-contri..., generate-stats...., kfp-operator.ip..., overview_with_..., overview_with_..., overview_with_..., and overview.ipynb. The file 'demo.pipeline' is highlighted. On the right, there is a 'Launcher' panel with three sections: 'demo-pipelines' (Notebook), 'Console' (Python on Kubernetes with, Python on Kubernetes, Python on Kubernetes with, Scala on Kubernetes, Spark - Python (Kubernetes, Spark - Scala (Kubernetes)), and 'Other' (Terminal, Text File, Markdown File, Pipeline Editor, Python File, Contextual Help). The 'Launcher' tab is selected at the bottom.

JupyterLab

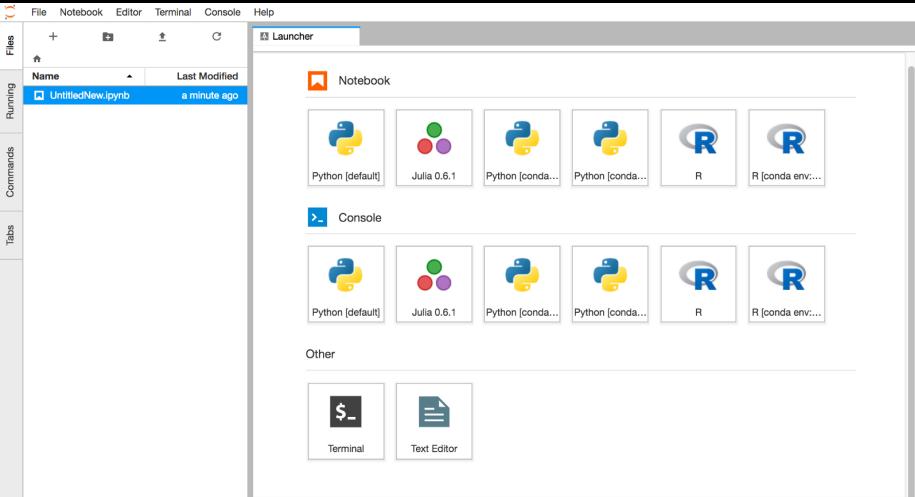


JupyterLab is the next generation UI for the Jupyter Ecosystem.

Bring all the previous improvements into a single unified platform plus more!

Provides a modular, extensible architecture

Retains backward compatibility with the old notebook we know and love





Notebook Pipelines editor

Elyra provides a visual editor for building Notebook-based AI pipelines, enabling the conversion of multiple notebooks into batch jobs or workflows.

Notebook as batch jobs

Elyra extends the notebook UI to simplify the submission of notebooks as a batch job for model training

Hybrid runtime support

It simplifies the task of running the notebooks interactively on cloud machines, improving productivity by leveraging the power of cloud-based resources

Python script execution

Exposes Python Scripts as first-class citizens allowing users to locally edit their scripts and execute them against local or cloud-based resources seamlessly.

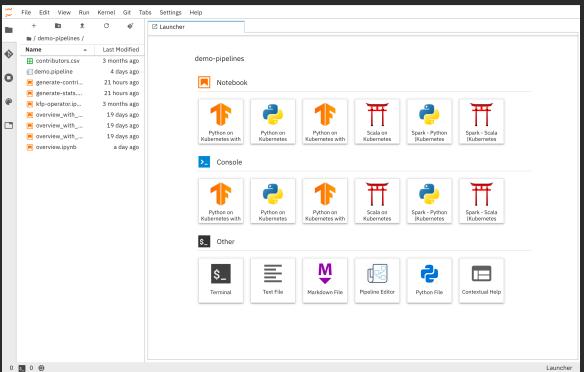
Versioning using git

Simplify tracking changes, enabling better sharing among teammates

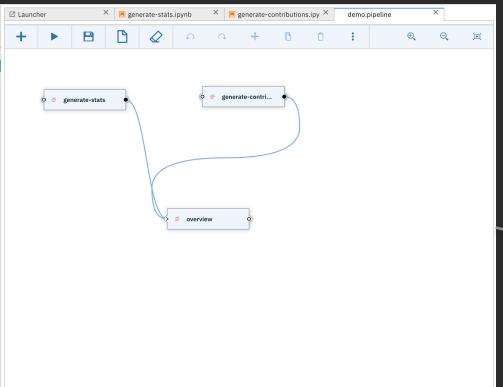
IBM Developer

© 2020 IBM Corporation

JupyterLab Extensions



Notebook Pipelines

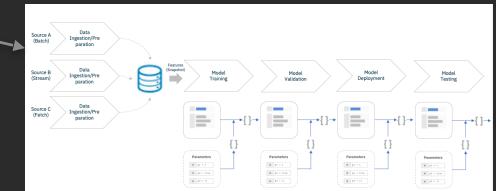


bit.ly/0901-ELYRA

Hybrid Runtime Support



Bring your own
Notebook



Data Asset eXchange

A place to find curated **free** and **open** datasets under *open data licenses*

ibm.biz/data-exchange

Data Asset eXchange

Explore useful and relevant data sets for enterprise data science

Learn More →

What's New →

Get Involved →

Dataset | CSV

NOAA Weather Data -
JFK Airport

September 12, 2019

Dataset | IOB format

Groningen Meaning
Bank - Modified

May 14, 2020

Dataset | CSV

Fashion-MNIST

September 12, 2019

Dataset | JPG, JSON

PubLayNet

October 25, 2019

Dataset | WAV

TensorFlow Speech
Commands

March 17, 2020

Dataset | PNG, JSON

PubTabNet

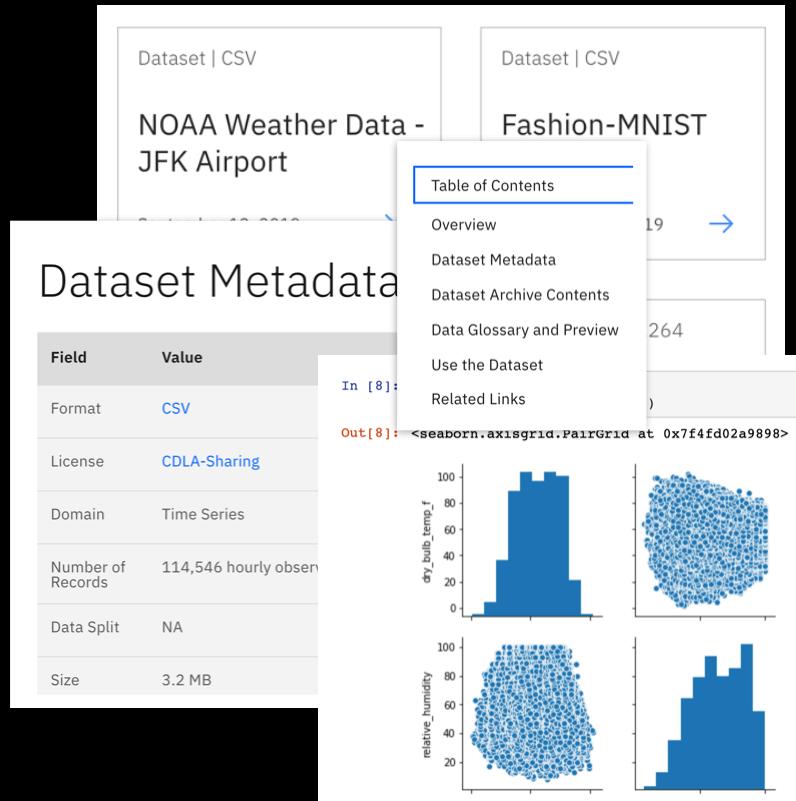
November 11, 2019

Data Asset eXchange

Standardized dataset formats and metadata

Ready for use in enterprise AI applications

Many data sets include starter notebooks (cleansing, data exploration, analysis)



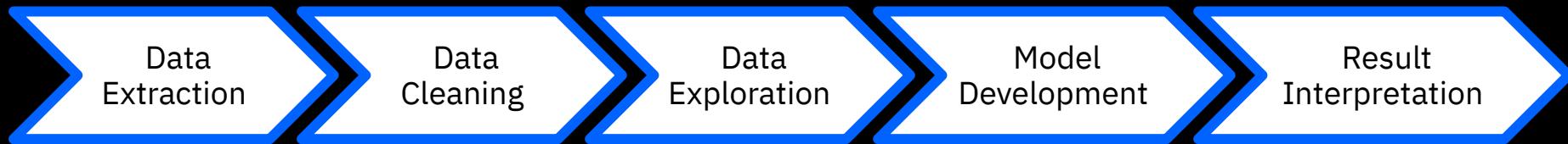
Let me show you around ...

ibm.biz/data-exchange

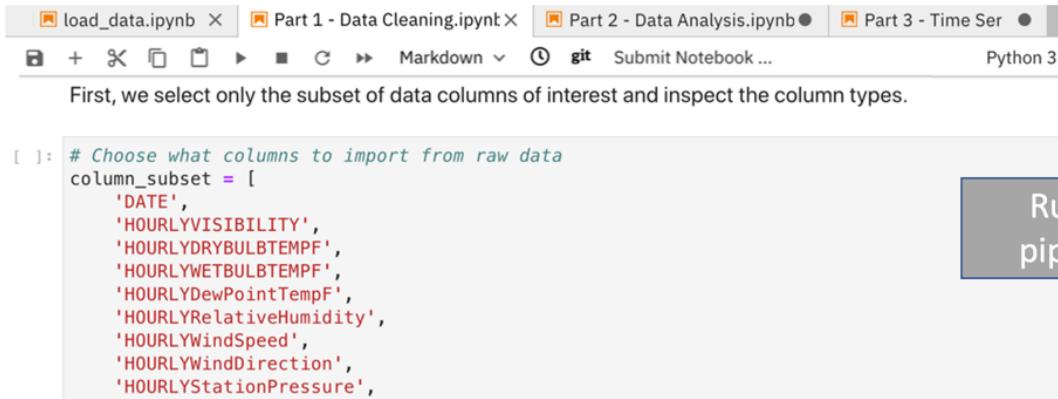
Sign up for IBM cloud: ibm.biz/BdqVxW



Data Science Pipeline

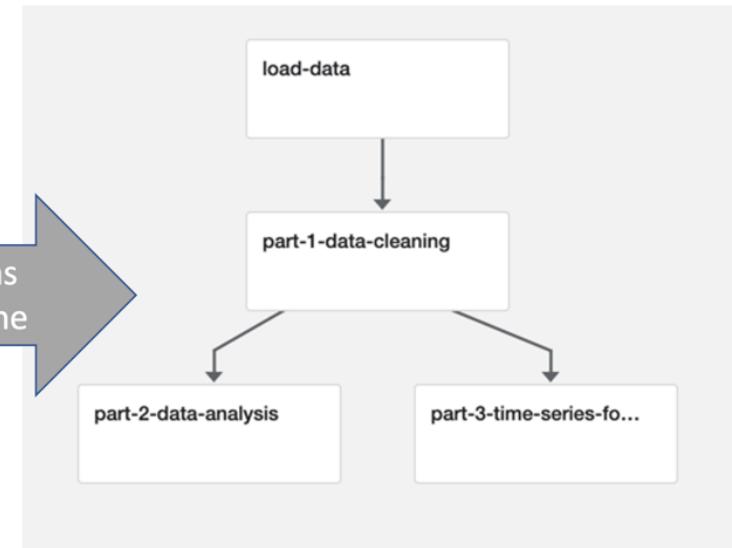


Creating notebook pipelines using Elyra and KubeFlow pipelines



```
[ ]: # Choose what columns to import from raw data
column_subset = [
    'DATE',
    'HOURLYVISIBILITY',
    'HOURLYDRYBULBTEMPF',
    'HOURLYWETBULBTEMPF',
    'HOURLYDewPointTempF',
    'HOURLYRelativeHumidity',
    'HOURLYWindSpeed',
    'HOURLYWindDirection',
    'HOURLYStationPressure',
```

Run as pipeline



Getting Started

What are the pre-requisites to run?

1. NodeJS 12+
2. Python 3.X
3. Anaconda (optional)
4. KubeFlow installation (optional)

Install Elyra

To install Elyra:

```
$ pip install elyra==1.1.0 && jupyter lab build
```

Or:

```
$ pip install --upgrade elyra && jupyter lab build
```

To verify installation:

```
$ jupyter serverextension list
```

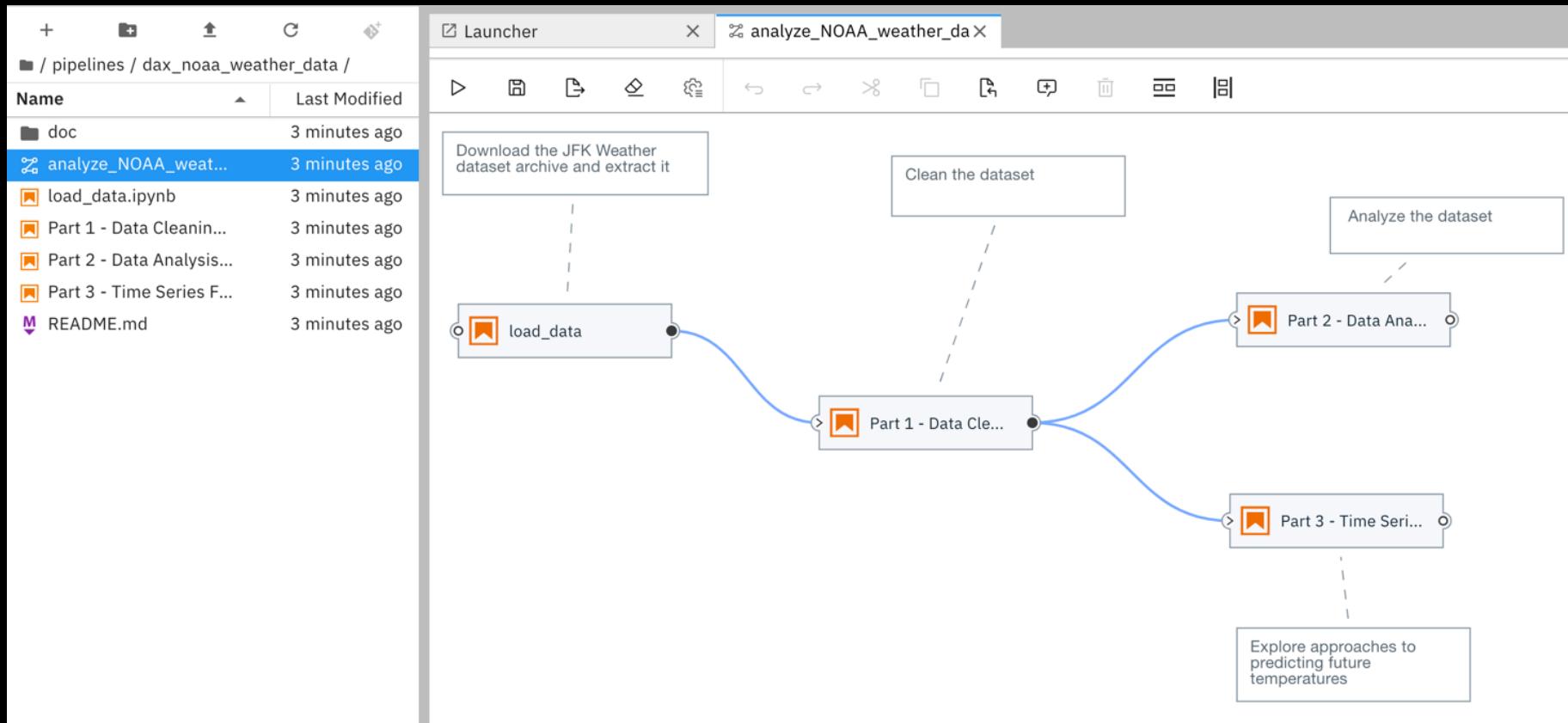
And

```
$ jupyter labextension list
```

Starting Elyra:

```
$ jupyter lab
```

Running notebooks as a pipeline



Let's run a pipeline locally and check out the notebooks!

Define Run Variables

The screenshot shows a configuration dialog for a Jupyter notebook run. It includes fields for:

- Filename:** load_data.ipynb
- Runtime Image (docker image used as execution environment):** Pandas
- File Dependencies:** Local file dependencies that need to be copied to remote execution environment. One filename or extension wildcard (eg. *.py) per line.
- Environment Variables:** DATASET_URL=https://dax-cdn.cdn.appdomain.cloud/dax-noaa-weather-data-jfk-airport/1.1.4/noaa-weather-data-jfk-airport.tar.gz
- Output Files:** data/noaa-weather-data-jfk-airport/jfk_weather.csv

At the bottom are **Cancel** and **Save** buttons, with **Save** highlighted in blue.

Several variables need to define:

- Docker runtime Image, such as Pandas image, TensorFlow image (w/ GPU support), Anaconda, or PyTorch (with CUDA-devel or with CUDA-runtime)
- File Dependencies
- Environment Variables
- Output files

Quickly add code snippet

The screenshot shows the IBM Watson Studio interface. On the left, there is a vertical toolbar with various icons. The main area is titled "Add new Code Snippet". It has two input fields: "Name (required)" and "Description (optional)". Below these is a dropdown menu labeled "Language (required)" with the placeholder "(No selection) ▾". A search bar labeled "Filter..." is positioned above a list of languages: Python, Java, R, and Scala. At the bottom of the dialog is a "Save & Close" button.

Code Snippets +

analyze_NOAA_weather_da X New Code Snippet X Part 1 - Data Cleaning.ipynb X Part 2 - Data Analysis.ipynb X Part 3 - Time Series Forecasting.ipynb X

Add new Code Snippet

Name (required)

Description (optional)

Language (required)

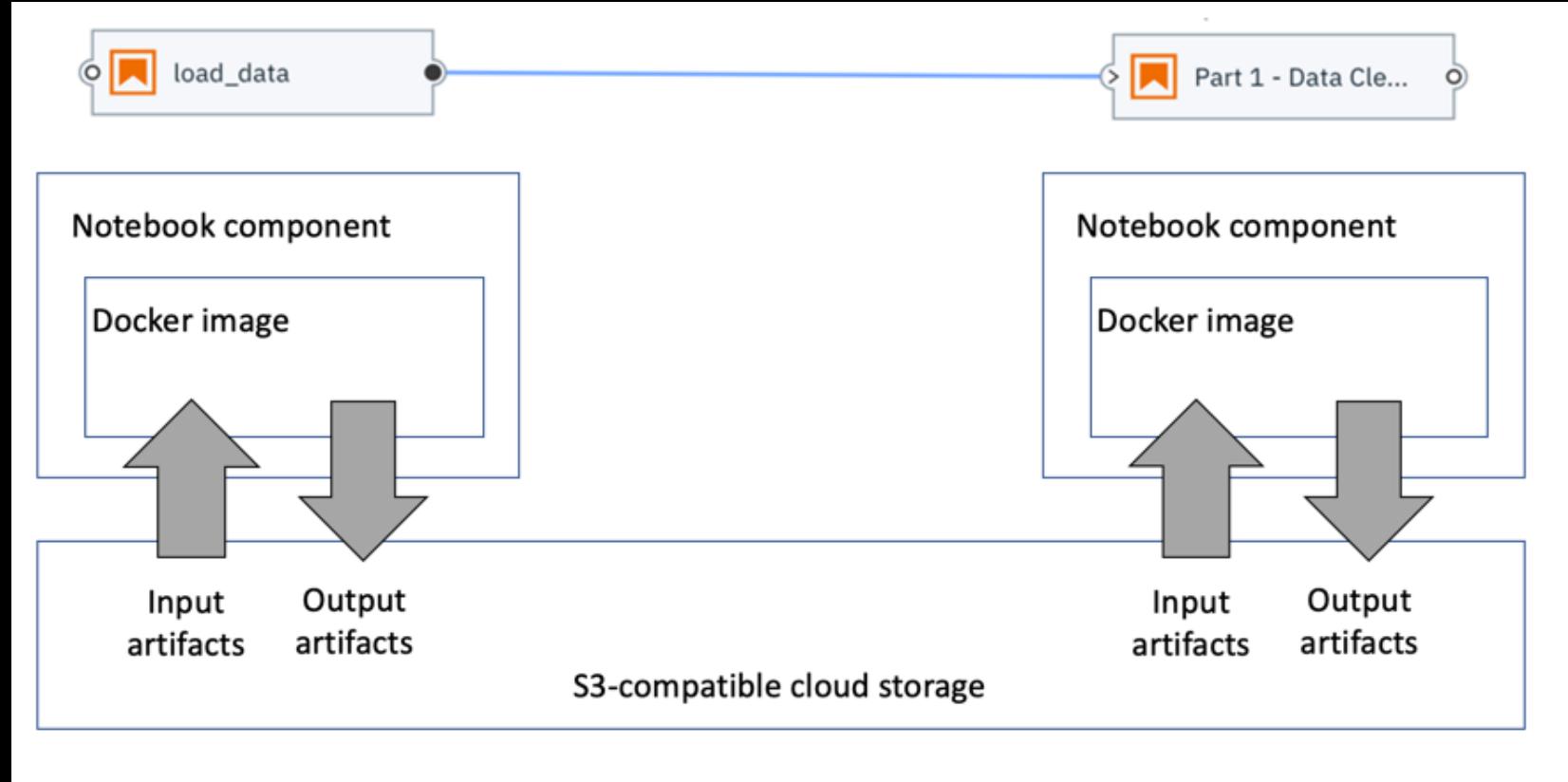
(No selection) ▾

Filter...

- Python
- Java
- R
- Scala

Save & Close

Configuring notebook nodes



If you want to run your pipeline on KubeFlow...

Configure a Kubeflow Pipeline runtime

Launcher × ⚙ analyze_NOAA_weather_da × My Kubeflow Pipeline - yukli ×

Edit "My Kubeflow Pipeline - yukked1"

Name (required)	Description (optional)
My Kubeflow Pipeline - <input type="text"/>	<input type="text"/>
Kubeflow Pipelines API Endpoint (required)	Cloud Object Storage Endpoint (required)
http:// <input type="text"/>	http:// <input type="text"/>
Cloud Object Storage Username (required)	Cloud Object Storage Password (required)
<input type="text"/> 
Cloud Object Storage Bucket Name (required)	
<input type="text"/> yiwenli	
<input type="button" value="Save & Close"/>	



Kubeflow

Manage runtime config using CLI

```
$ elyra-metadata install runtimes --schema_name=kfp \
--name=kfp_dev_instance \
--display_name="KFP dev instance" \
--api_endpoint=http://.../pipeline \
--cos_endpoint=http://... \
--cos_username=... \
--cos_password=... \
--cos_bucket=...

$ elyra-metadata list runtimes
Available metadata instances for runtimes (includes invalid):
Schema  Instance      Resource
-----  -----
kfp     kfp_dev_instance /Users/.../kfp_dev_instance.json
```

```
$ elyra-metadata list runtimes --json
[
  {
    "name": "kfp_dev_instance",
    "display_name": "KFP dev instance",
    "metadata": {
      "api_endpoint": "http://.../pipeline",
      "cos_endpoint": "http://...",
      "description": "...",
      "cos_username": "...",
      "cos_password": "...",
      "cos_bucket": "..."
    },
    "schema_name": "kfp",
    "resource": "/Users/.../kfp_dev_instance.json"
  }
]
$ elyra-metadata remove runtimes --name=kfp_dev_instance
Metadata instance '...' removed from namespace 'runtimes'.
```

Monitor a notebook run

The screenshot shows a Jupyter Notebook interface with a sidebar on the left containing icons for file operations, a search bar, and a help section. The main area has a header with "File", "Edit", "View", "Run", "Kernel", "Git", "Tabs", "Settings", and "Help". Below the header, the "Runtimes" section is displayed, featuring a "My Kubeflow Pipeline - yukked1" entry with edit and delete icons. A collapsed section below it lists "Kubeflow Pipelines UI" and "Cloud Object Storage" with their respective URLs.

Runtimes

My Kubeflow Pipeline - yukked1

Kubeflow Pipelines UI
[http://\[REDACTED\]/pipeline/](http://[REDACTED]/pipeline/)

Cloud Object Storage
[http://\[REDACTED](http://[REDACTED)

Let's view the results on KubeFlow!

Pipelines

Experiments

Artifacts

Executions

Archive

Documentation

Github Repo

AI Hub Samples

Experiments

+ Create run

+ Create experiment

Compare runs

Clone run

All experiments

All runs

Filter experiments



Experiment name	Description	Last 5 runs
▶ aaaa-0825120733		✓
▶ 11_test2-0825120320		✓
▶ 11_validation-0825112943		✓
▶ yiwen-0824164826		✓
▶ yiwen-0824141355		✓
▶ aaa2-0824114351		✓
▶ aaa1-0824114116		!
▶ aaa-0824113956		!
▶ aaa-0824104836		✓
▶ Default	All runs created without specifying an experiment will be grouped here.	

Pipelines

Experiments

Artifacts

Executions

Archive

Documentation

Github Repo

AI Hub Samples

Build commit: 9c16e12

Report an Issue

Experiments > yiwen-0824164826

← 0824164826

Retry Clone run Terminate Archive

Graph Run output Config

```

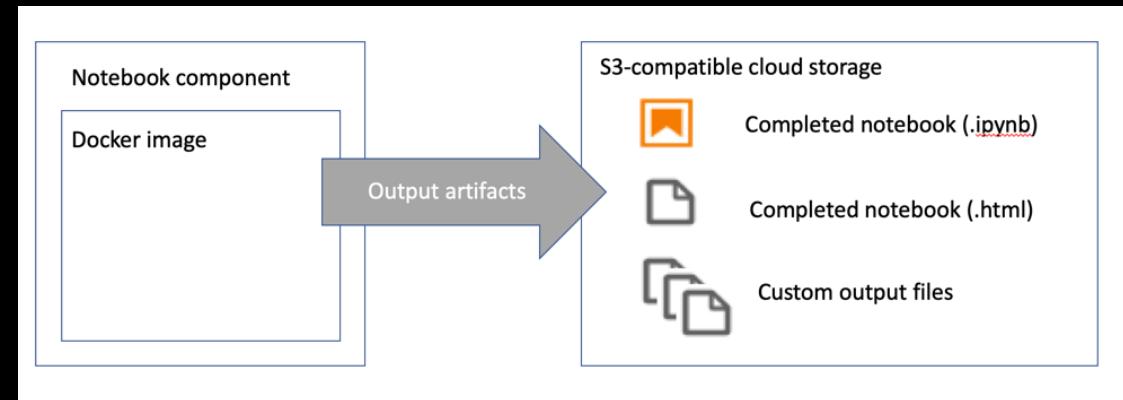
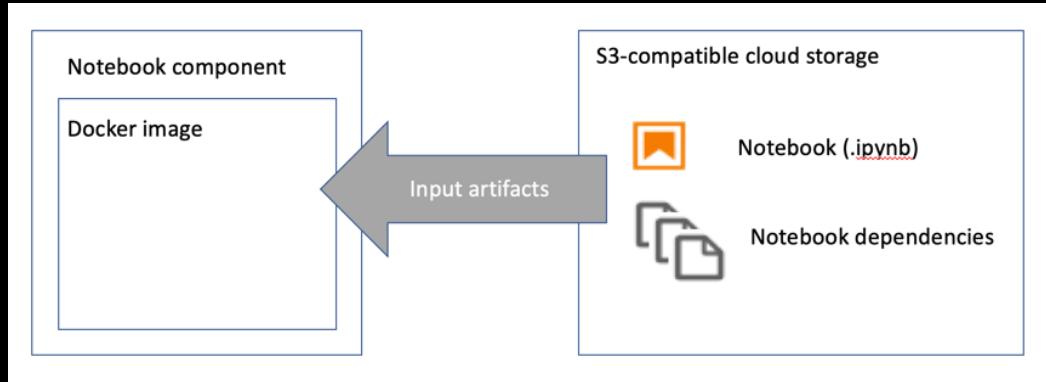
graph TD
    A[load-data] --> B["part-1-data-clean..."]
    B --> C["part-2-data-analy..."]
    B --> D["part-3-time-serie..."]
  
```

lambda-k4w7d-3201965441

Artifacts	Input/Output	ML Metadata	Volumes	Manifest	Logs	Pod	Events
210 ipykernel==0.1.2							
211 testpath==0.4.4							
212 textwrap3==0.9.2							
213 toml==0.10.1							
214 tornado==6.0.4							
215 tqdm==4.48.2							
216 traitlets==4.3.3							
217 typed-ast==1.4.1							
218 urllib3==1.25.9							
219 wccwidth==0.2.5							
220 webencodings==0.5.1							
221 zipp==3.1.0							
222 /							
223 load_data.ipynb							
224 tar: Removing Leading '/' from member names							
225 Package not found. Installing ipykernel package with version 5.3.0...							
226 Package not found. Installing ipython package with version 7.15.0...							
227 Package not found. Installing ipython-genutils package with version 0.2.0...							
228 Package not found. Installing jupyter-client package with version 6.1.6...							
229 Package not found. Installing jupyter-core package with version 4.6.3...							
230 Package not found. Installing minio package with version 5.0.10...							
231 Package not found. Installing nbclient package with version 0.4.1...							
232 Package not found. Installing nbconvert package with version 5.6.1...							
233 Package not found. Installing nbformat package with version 5.0.7...							
234 Package not found. Installing papermill package with version 2.1.2...							
235 Package not found. Installing prompt-toolkit package with version 3.0.5...							
236 Package not found. Installing pyzmq package with version 19.0.1...							
237 Package not found. Installing requests package with version 2.23.0...							
238 Package not found. Installing tornado package with version 6.0.4...							
239 Package not found. Installing traitlets package with version 4.3.3...							
240 Package not found. Installing urllib3 package with version 1.25.9...							
241 Package Installation Complete.....							
242 Parsing Arguments.....							
243 Get file load_data-c88d9c0b-a5d5-45ab-88d6-eb6ce24ffdbb.tar.gz from bucket yiwenli							
244 Processing dependencies.....							
245 TAR Archive pulled from Object Storage.							
246 Unpacking.....							
247 Unpacking Complete.							
248 Executing notebook through Papermill: load_data.ipynb ==> load_data-output.ipynb							
249 Executing: 0% 0/8 [00:00<?, ?cell/s] Executing: 12% 1/8 [00:00<00:06, 1.14cell/s] Executing: 75%							
250 Converting from ipynb to html....							
251 Uploading Result Notebook back to Object Storage							
252 Uploading file load_data-output.ipynb as load_data.ipynb to bucket yiwenli							
253 Uploading file load_data.html as load_data.html to bucket yiwenli							
254 Processing outputs.....							
255 Uploading file noaa-weather-data-jfk-airport/jfk_weather.csv as data/noaa-weather-data-jfk-airport/jfk_weather.csv to bucket yiwe							
256 Upload Complete.							

Runtime execution graph. Only steps that are currently running or have

How KubeFlow pipeline works?



Pipeline outputs

Name		Size	Last Modified	⋮
data/				
 Part 3 - Time Series Forecasting.html		564.73 KB	Aug 24, 2020 4:50 PM	⋮
 Part 3 - Time Series Forecasting.ipynb		956.79 KB	Aug 24, 2020 4:50 PM	⋮
 Part 2 - Data Analysis.html		3.62 MB	Aug 24, 2020 4:50 PM	⋮
 Part 2 - Data Analysis.ipynb		3.81 MB	Aug 24, 2020 4:50 PM	⋮
 Part 1 - Data Cleaning.html		348.84 KB	Aug 24, 2020 4:49 PM	⋮
 Part 1 - Data Cleaning.ipynb		110.26 KB	Aug 24, 2020 4:49 PM	⋮
 load_data.html		274.55 KB	Aug 24, 2020 4:49 PM	⋮
 load_data.ipynb		7.20 KB	Aug 24, 2020 4:49 PM	⋮
 Part 3 - Time Series Forecasting-b00e4654-a2b0-417c-8f93-8a03bec95945.tar.gz		9.59 KB	Aug 24, 2020 4:48 PM	⋮
 Part 2 - Data Analysis-982e672a-4ae5-4608-bcb0-ce309868415a.tar.gz		4.62 KB	Aug 24, 2020 4:48 PM	⋮
 Part 1 - Data Cleaning-e07e1b7f-568b-4bc3-9fc6-da372fd58daf.tar.gz		7.48 KB	Aug 24, 2020 4:48 PM	⋮
 load_data-c88d9c0b-a5d5-45ab-88d6-eb6ce24ffdbb.tar.gz		1.69 KB	Aug 24, 2020 4:48 PM	⋮

Get your KubeFlow

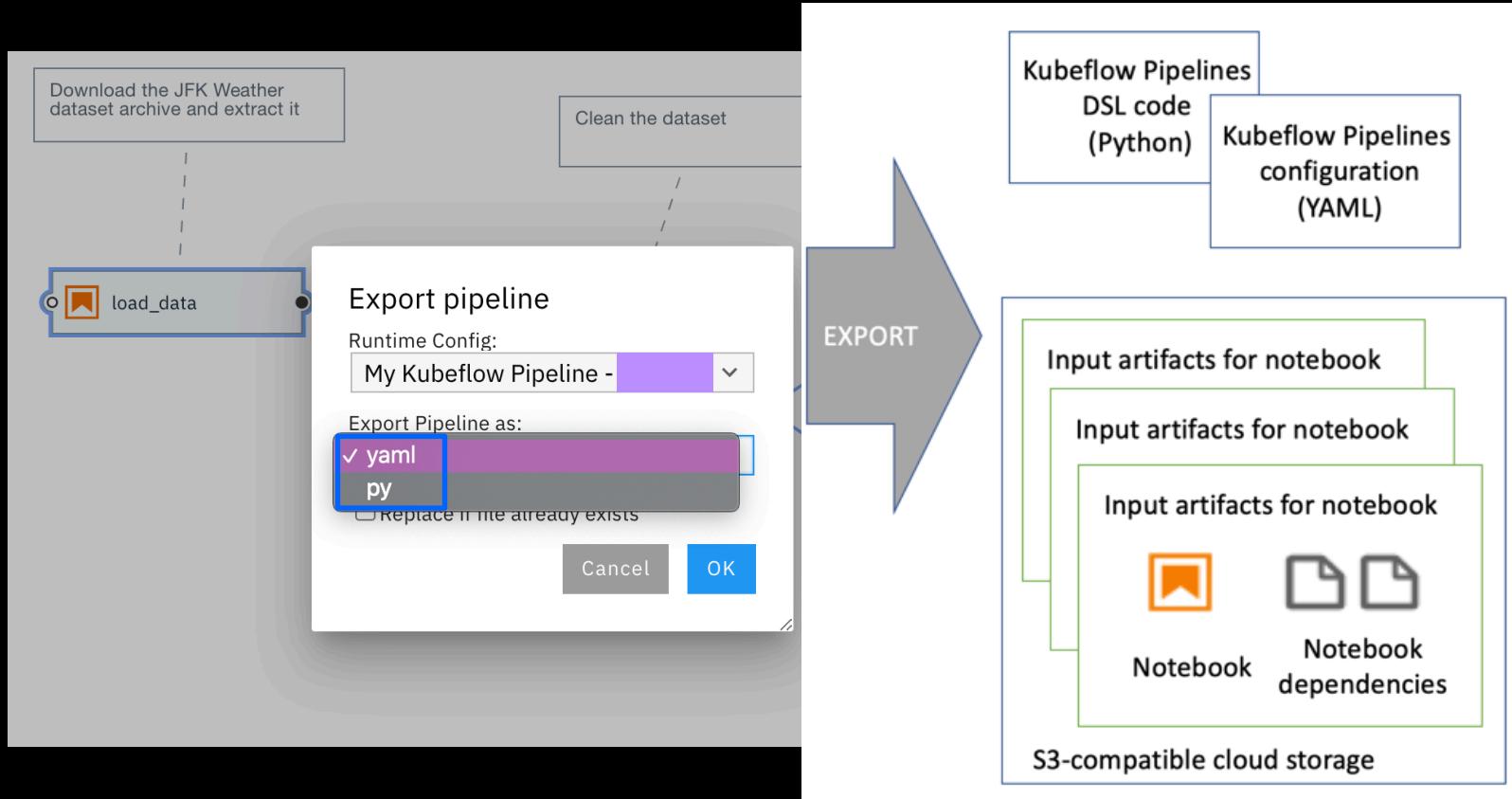
Run KubeFlow on IBM Cloud

<https://www.kubeflow.org/docs/ibm/>

See this tutorial to run with KubeFlow

https://github.com/elyra-ai/examples/tree/master/pipelines/hello_world

Export a pipeline



Ways to run pipelines

Try JupyterLab and Elyra on pre-built docker image:

```
$ docker run -it -p 8888:8888 elyra/elyra:1.0.0 jupyter lab --debug
```

Or you can use Binder:

<https://mybinder.org/v2/gh/elyra-ai/elyra/v1.0.0?urlpath=lab/tree/binder-demo>

Or you can install Elyra:

<https://github.com/elyra-ai/elyra#installation>

ELYRA & PYTHON SUPPORT

Fork me at: github.com/elyra-ai/elyra



The screenshot shows the ELYRA Python support interface. On the left is a file browser with a file named "PANDAS.PY". The main area has two tabs: "Launcher" and "pandas.py". The "Launcher" tab shows a Python script:

```
import io
def df_from_url(url)
import time
import requests
import pandas as pd
def df_from_url(url):
    data = requests.get(url).content
    df = pd.read_csv(io.StringIO(data.decode('utf-8')))
    return df
import time
time.sleep(5)
df = df_from_url('http://samplecsvs.s3.amazonaws.com/Sacramentoalestatetransactions.csv')
df.groupby('zip')['price'].mean()
```

The "pandas.py" tab shows the Python Console Output:

```
[ ]: zip
      95603   405890.800000
      95608   295684.750000
      95610   226436.285714
      95614   300000.000000
      95619   216033.000000
      ...
      95838   149461.351351
      95841   213806.142857
      95842   143281.772727
      95843   232496.393939
      95864   364400.000000
Name: price, Length: 68, dtype: float64
```

At the bottom, the status bar shows "0 0 Python" and "Ln 1, Col 1 Spaces: 4 pandas.py".

ELYRA & CODE - SNIPPETS

Fork me at: github.com/elyra-ai/elyra



File Edit View Run Kernel Git Tabs Settings Help

Launcher pandas.py Untitled.ipynb Python 3

Code Snippets

- [scala] Spark - Bank Scenario
- [scala] Spark - Configuration details
- [Lua] aa
- [python] Matplotlib Configuration
- from __future__ import print_function, division
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
- [python] Matplotlib simple graph
- Matplotlib simple graph
- # Make the plot
bp_x = np.linspace(0, 2*np.pi, num=40, endpoint=True)
bp_y = np.sin(bp_x)
- # Make the plot
plt.plot(bp_x, bp_y, linewidth=3, linestyle='--', color='blue', label=r'Legend label sin(x)')
plt.xlabel(r'Description of \$x\$ coordinate (units)')
plt.ylabel(r'Description of \$y\$ coordinate (units)')
plt.title(r'Title here (remove for papers)')
plt.xlim(0, 2*np.pi)
plt.ylim(-1.1, 1.1)
plt.legend(loc='lower left')
plt.show()

0 1 Python 3 | Idle Mode: Command Ln 1, Col 1 Untitled.ipynb

The screenshot shows the ELYRA interface. On the left, there's a sidebar titled 'Code Snippets' containing various code examples categorized by language and type. Two arrows point from specific snippets in this sidebar to the main workspace. One arrow points from the 'Matplotlib simple graph' snippet to the code cell in the center, which contains the code for generating a sine wave plot. Another arrow points from the same snippet to the resulting plot on the right. The main workspace consists of two code cells and a plot area. The top code cell (cell 4) contains imports for print_function, division, numpy, matplotlib, and pyplot, along with the %matplotlib inline magic command. The bottom code cell (cell 5) contains the code to generate a sine wave plot, including labels for the x and y axes, a title, and a legend. The resulting plot shows a blue dashed sine wave on a grid with x-axis ticks at 0, 1, 2, 3, 4, 5, 6 and y-axis ticks from -1.00 to 1.00 in increments of 0.25. A legend in the bottom-left corner identifies the series as 'Legend label sin(x)'.

Get involved with Elyra!

- Open an enhancement request
- Open an issue
- Contributing to Elyra!

Related Links

Slides: bit.ly/0901-ELYRA

Elyra Github: <https://github.com/elyra-ai/elyra>

Elyra Examples: <https://github.com/elyra-ai/examples>

Elyra Tutorial: https://github.com/elyra-ai/examples/tree/master/pipelines/hello_world

DAX Asset eXchange: <http://ibm.biz/data-exchange>

Elyra demo Github: https://github.com/elyra-ai/examples/tree/master/pipelines/dax_noaa_weather_data

Sign up for IBM Cloud: <https://ibm.biz/BdqVxW>

Model Asset eXchange: <http://ibm.biz/model-exchange>

My Contact

Email: yiwen.Li@ibm.com

Linkedin: <https://www.linkedin.com/in/yiwen-li-47a019119/>

GitHub: yil532

THANK YOU!

