

How to build a financial model with KubeFlow

葉信和 / Hsin-Ho Yeh
Software Engineer / CEO @ 信誠金融科技
hsinho.yeh@footprint-ai.com

Download Slides

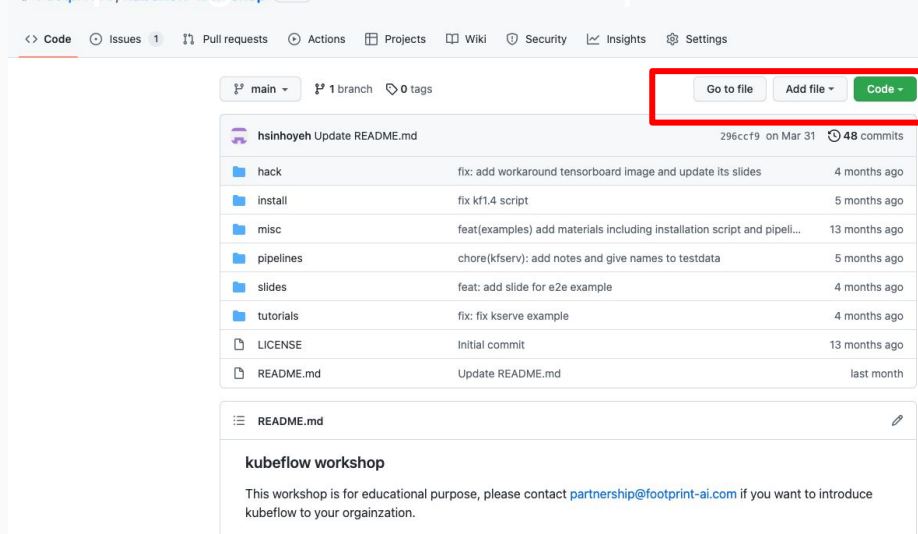
<https://reurl.cc/yrkGrE>



Materials

git clone

<https://github.com/FootprintAI/kubeflow-workshop>



The screenshot shows the GitHub repository page for `FootprintAI/kubeflow-workshop`. The repository is on the `main` branch. The `Code` button is highlighted with a red box. Below the repository name, there is a table of files and folders.

File/Folder	Description	Commit
hack	fix: add workaround tensorboard image and update its slides	4 months ago
install	fix kf1.4 script	5 months ago
misc	feat(examples) add materials including installation script and pipeli...	13 months ago
pipelines	chore(kfserv): add notes and give names to testdata	5 months ago
slides	feat: add slide for e2e example	4 months ago
tutorials	fix: fix kserve example	4 months ago
LICENSE	Initial commit	13 months ago
README.md	Update README.md	last month

The `README.md` file is selected, showing the following content:

```
kubeflow workshop
```

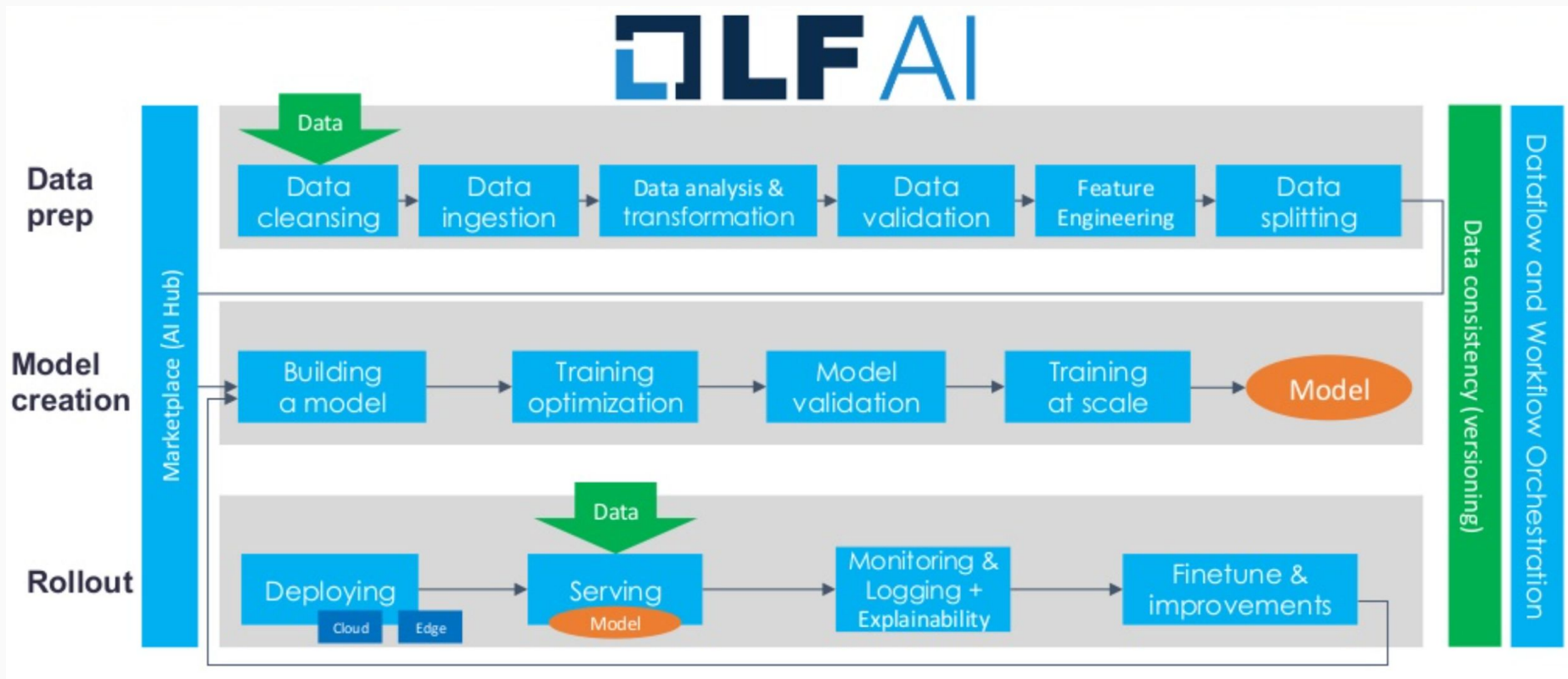
This workshop is for educational purpose, please contact partnership@footprint-ai.com if you want to introduce kubeflow to your organization.

About me

- 2020 - Present at 信誠金融科技
 - Shrimping: A data-sharing platform
 - <https://get-shrimping.footprint-ai.com>
 - Tintin: a machine learning platform for everyone
 - <https://get-tintin.footprint-ai.com>
- 2016 - 2020 at IglooInsure (16M+ in series A+ 2020)
 - Provide digital insurance for e-economic world
 - Funded in KUL, Headquartered in Singapore
 - First employee/ Engineering Lead / Regional Head/ Chief Engineer
- 2013 - 2016 at Studio Engineering @ hTC
 - Principal Engineer on Cloud Infrastructure Team
- 2009 - 2012 at IIS @ Academia Sinica
 - Computer vision, pattern recognition, and data mining
- CS@CCU, CS@NCKU alumni

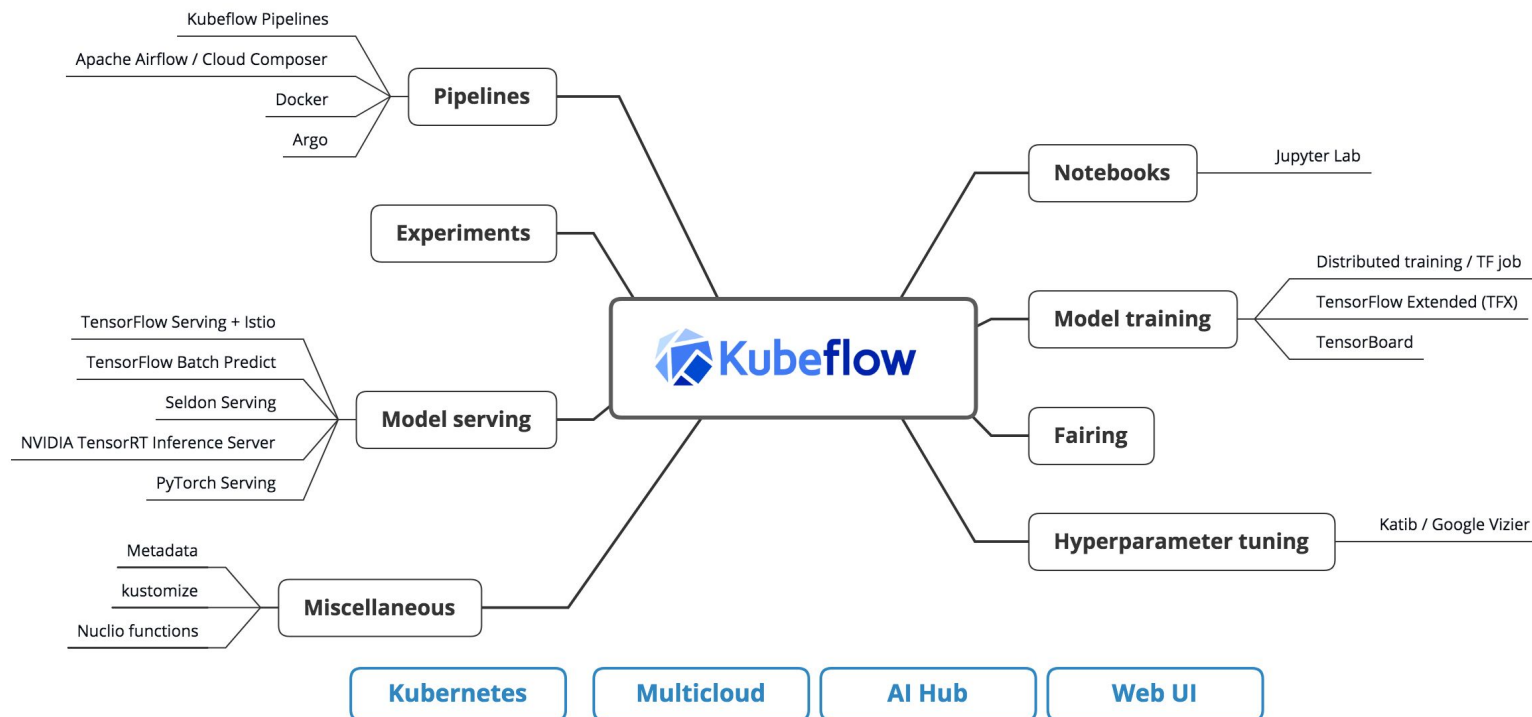


Real-world Machine Learning Application - End-to-End ML LifeCycle



Source: <https://www.slideshare.net/AnimeshSingh/advanced-model-inferencing-leveraging-kubeflow-serving-knative-and-istio-196096385>

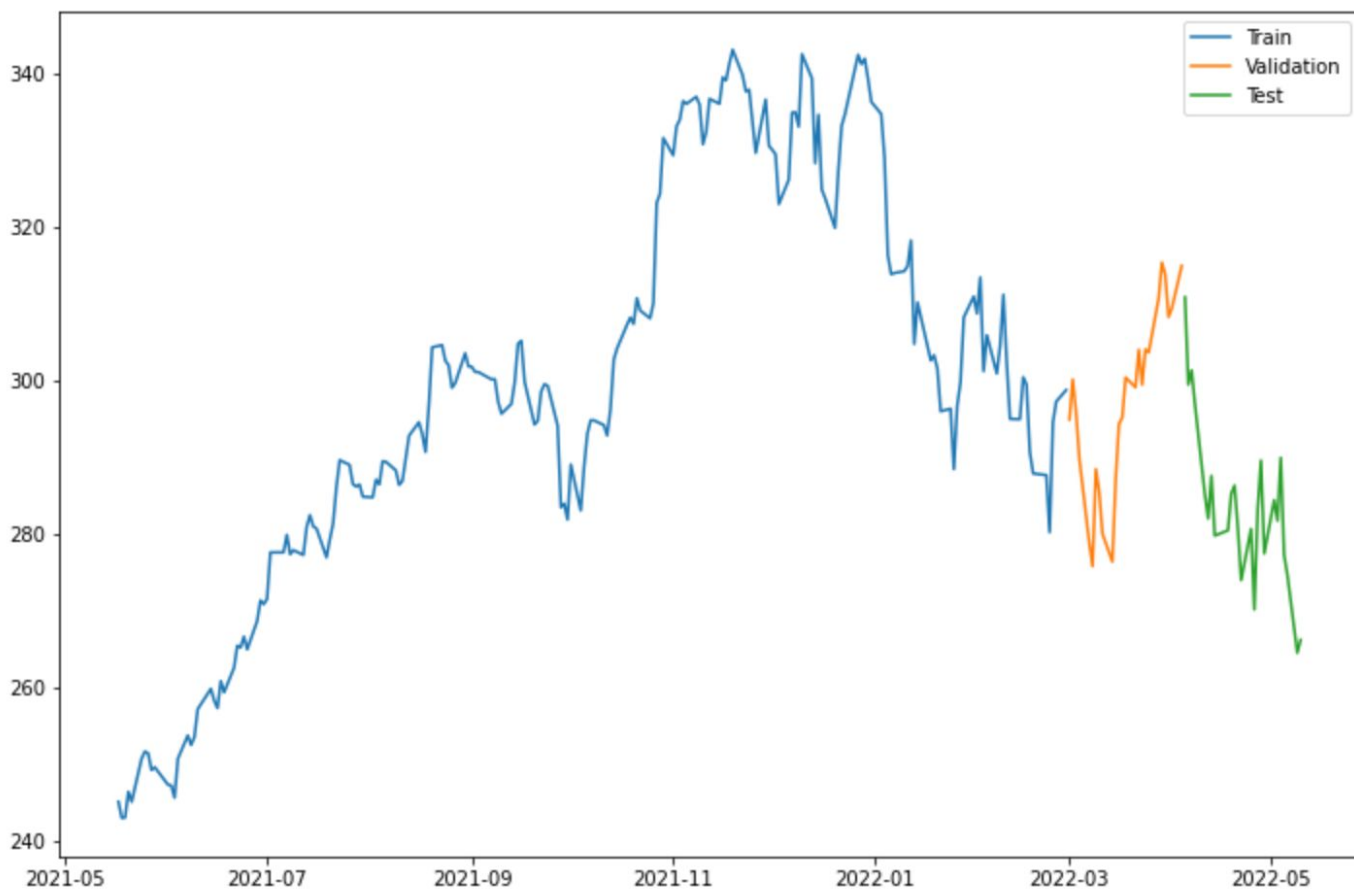
Kubeflow Architecture



Kubeflow In Financial Fields

Can LSTM model be used to predict
the stock price?

How to predict stock price for the next day?



What sequences means in a series?

Input: [300, 320, 310, 350, 390]

t0	300
t1	320
t2	310
t3	350
t4	390
t5	?

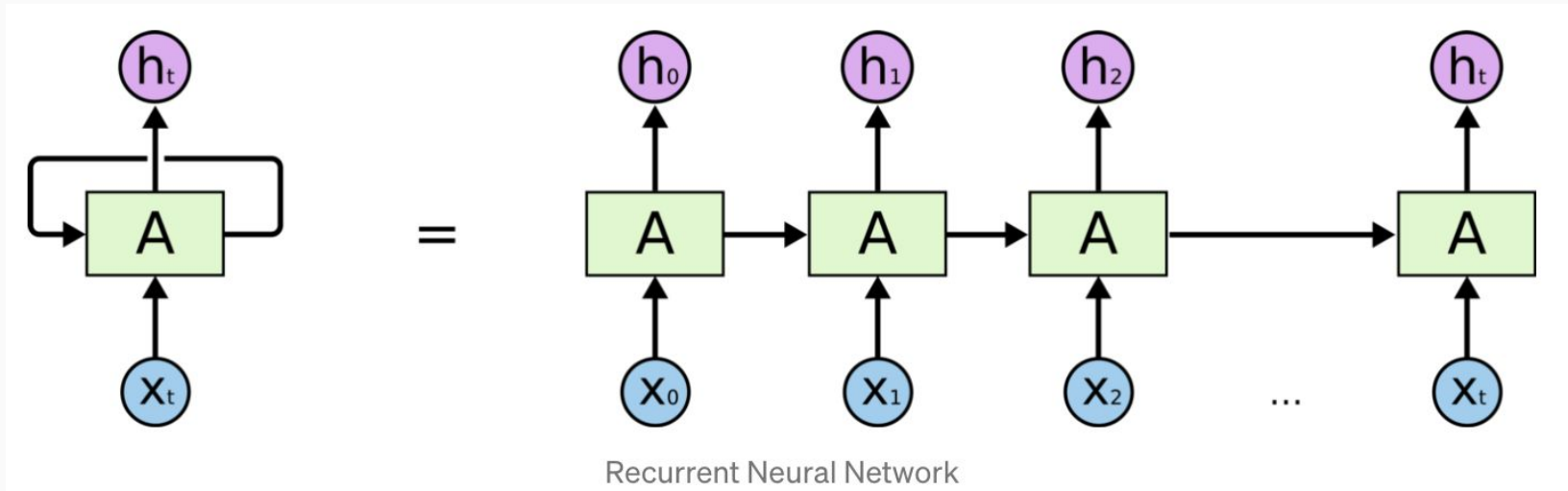
What sequences means in a sentence?

Input: I am a good guy

t0	I
t1	am
t2	a
t3	good
t4	guy
t5	?

What Is RNN (Recurrent Neural Network)?

Recurrent Neural Network (RNN) takes decisions on CURRENT (X_t) and PREVIOUS (X_{t-1}) inputs. Especially useful in topics including machine translation, speech recognition.

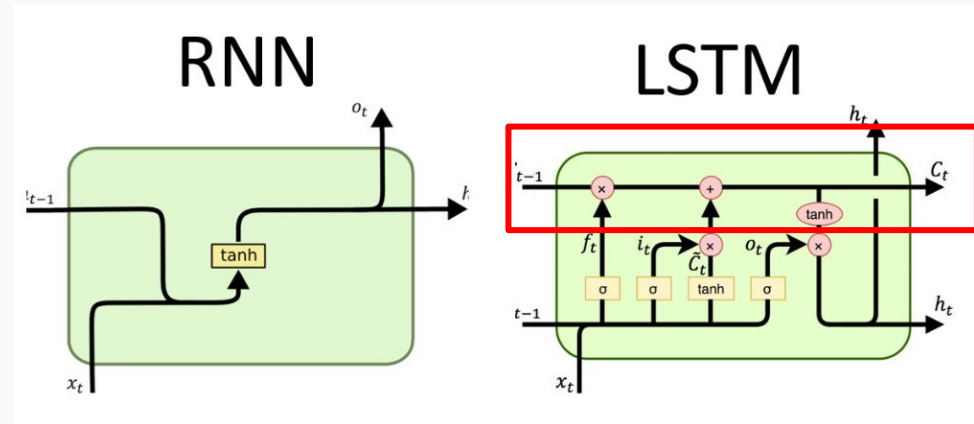


What Is LSTM (Long-short term memory)?

RNN only remember the latest things from X and it didn't remember(no memory) anything before at the beginning.

LSTM provides an information highway to let the neuron to selectively choose

1. forget from its memory (focus on the current inputs)
2. Listens to what information it added into memory (though information highway)




<https://towardsdatascience.com/introduction-to-recurrent-neural-network-27202c3945f3>

<https://www.quora.com/How-is-LSTM-different-from-RNN-In-a-layman-explanation>

How to formulate a sequence into a trainable dataset?

t0	t1	t2	t3	t4
300	320	310	350	390



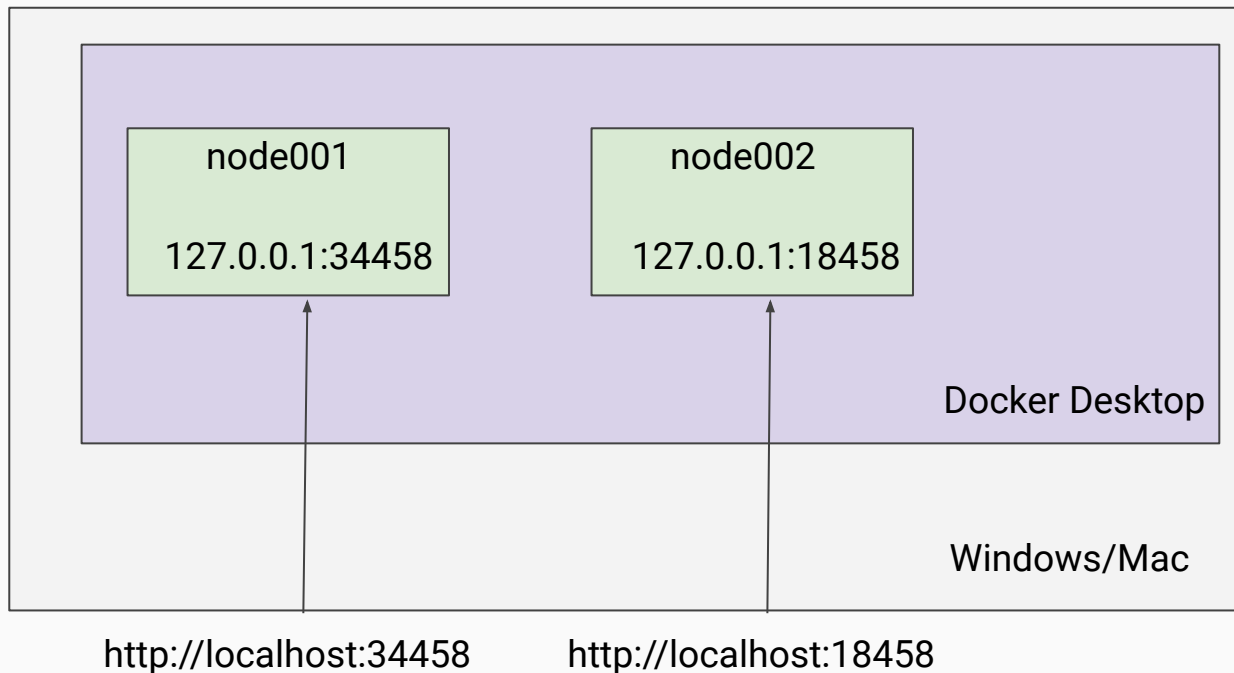
Windows size 3, we look at last three days data

Feature 1	Feature 2	Feature 3	Target
300	320	310	350
320	310	350	390

Environment Setup

Deployment Overview

Provisioning with multikf <https://github.com/FootprintAI/multikf/tree/main/docs>



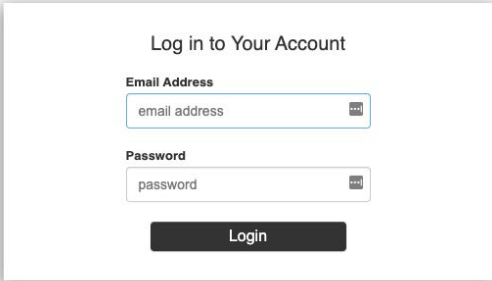
Deployment Overview: 4 steps setup

1. Download multikf
2. `./multikf add node001`
3. `./multikf connect kubeflow node001`
4. Open browser on the dedicated address

Wait! 所以我說那個帳號密碼呢?

Account: [user@example.com](#)

Password: 12341234



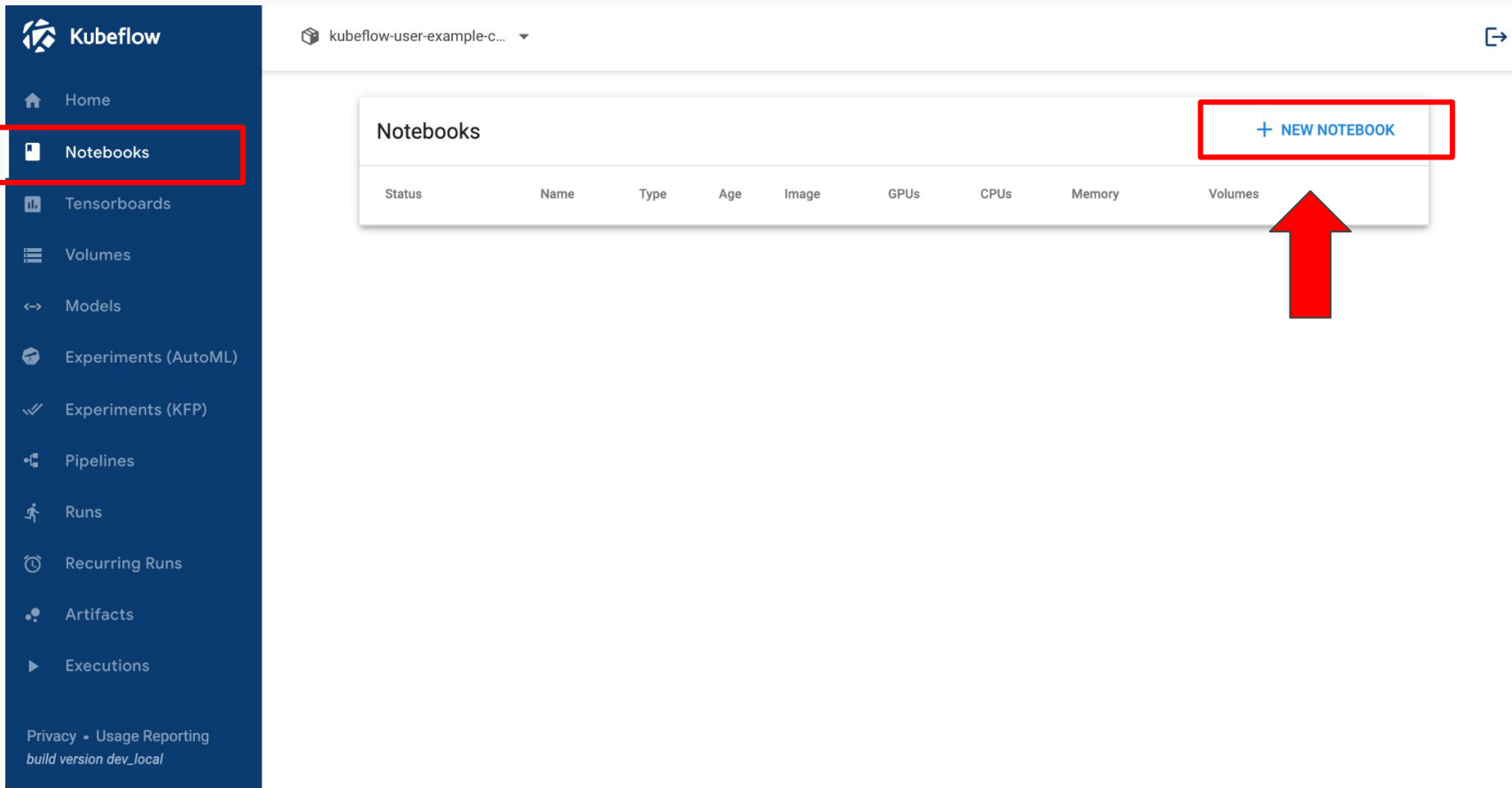
Log in to Your Account

Email Address

Password

Login

Step1: Use Notebook as Online IDE (1/3)



The screenshot displays the Kubeflow dashboard interface. On the left, a dark blue sidebar contains navigation links: Home, Notebooks, Tensorboards, Volumes, Models, Experiments (AutoML), Experiments (KFP), Pipelines, Runs, Recurring Runs, Artifacts, and Executions. The 'Notebooks' link is highlighted with a red rectangle. The main content area shows a 'Notebooks' section with a table header and a '+ NEW NOTEBOOK' button. The button is also highlighted with a red rectangle, and a large red arrow points to it from below. The table header includes columns for Status, Name, Type, Age, Image, GPUs, CPUs, Memory, and Volumes. The top right of the dashboard shows a user profile dropdown and a share icon.

Kubeflow

kubeflow-user-example-c...

Home

Notebooks

Tensorboards

Volumes

Models

Experiments (AutoML)

Experiments (KFP)

Pipelines

Runs

Recurring Runs

Artifacts

Executions


Privacy • Usage Reporting
build version dev_local

Notebooks

+ NEW NOTEBOOK

Status	Name	Type	Age	Image	GPUs	CPUs	Memory	Volumes
--------	------	------	-----	-------	------	------	--------	---------

Step1: Use Notebook as Online IDE (2/3)

 Kubeflow

Home

Notebooks

Tensorboards

Volumes

Models

Experiments (AutoML)

Experiments (KFP)

Pipelines

Runs


Recurring Runs

Artifacts

Executions

Manage Contributors

kubeflow-user-example-c...




Specify the name of the Notebook Server and the Namespace it will belong to.

Name

demo

Namespace

kubeflow-user-example-com

 Image

A starter Jupyter Docker Image with a baseline deployment of TensorFlow and JupyterLab ML packages

☐ Custom Image

jupyterlab

1

2

Image

j1r0q0g6/notebooks/notebook-servers/jupyter-tensorflow-full:v1.4

Advanced Options

CPU / RAM

Specify the total amount of CPU and RAM reserved by your Notebook Server. For CPU-intensive workloads, you can choose more than 1 CPU (e.g. 1.5).

Requested CPUs

0.5


Requested memory in Gi

1


Advanced Options

Specify a name and resources like CPU and Memory







Step1: Use Notebook as Online IDE (3/3)


 **Kubeflow**

[Home](#)
[Notebooks](#)
[Tensorboards](#)
[Volumes](#)
[Models](#)
[Experiments \(AutoML\)](#)
[Experiments \(KFP\)](#)
[Pipelines](#)
[Runs](#)
[Recurring Runs](#)
[Artifacts](#)

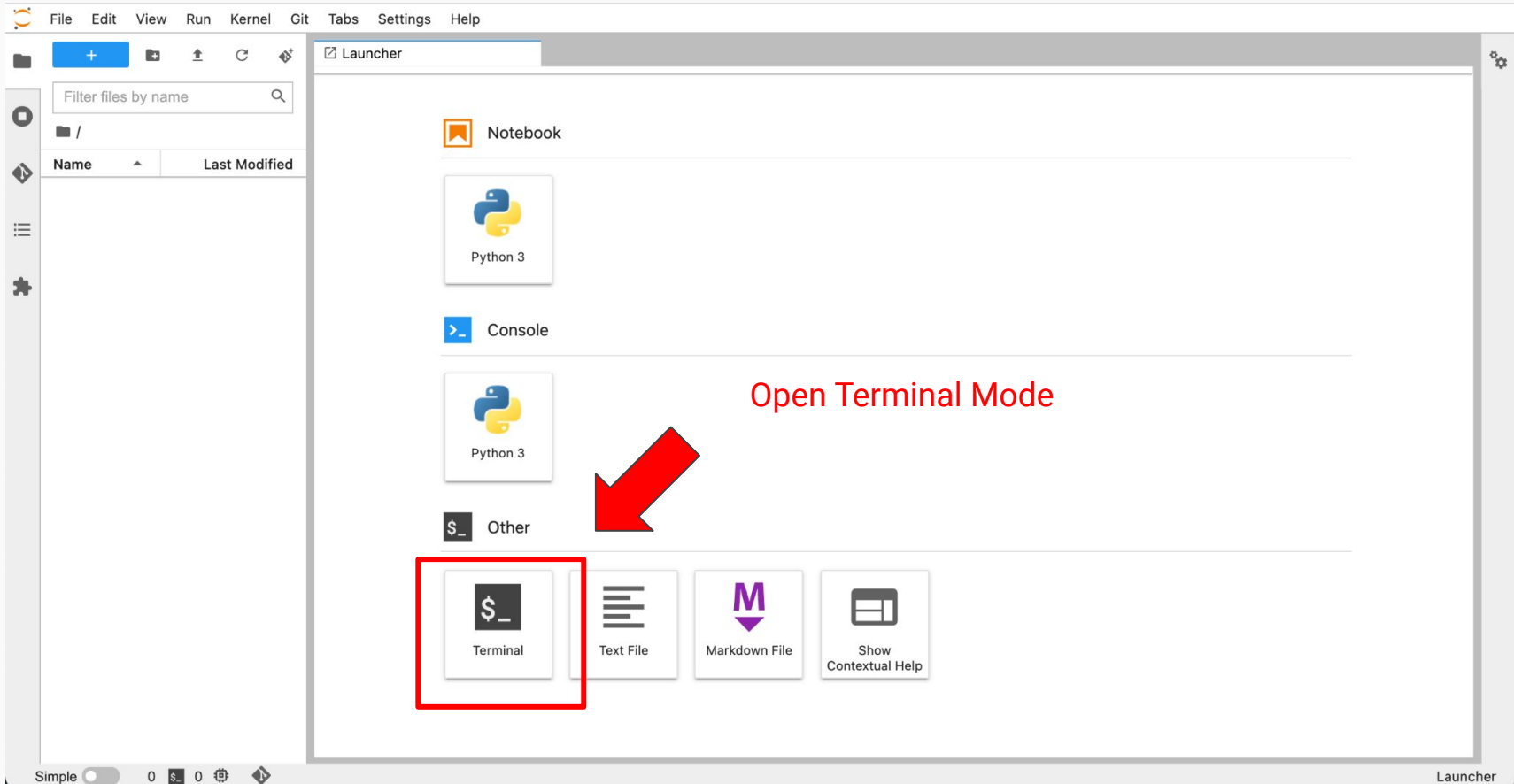
kubeflow-user-example-c... 

Notebooks [+ NEW NOTEBOOK](#)

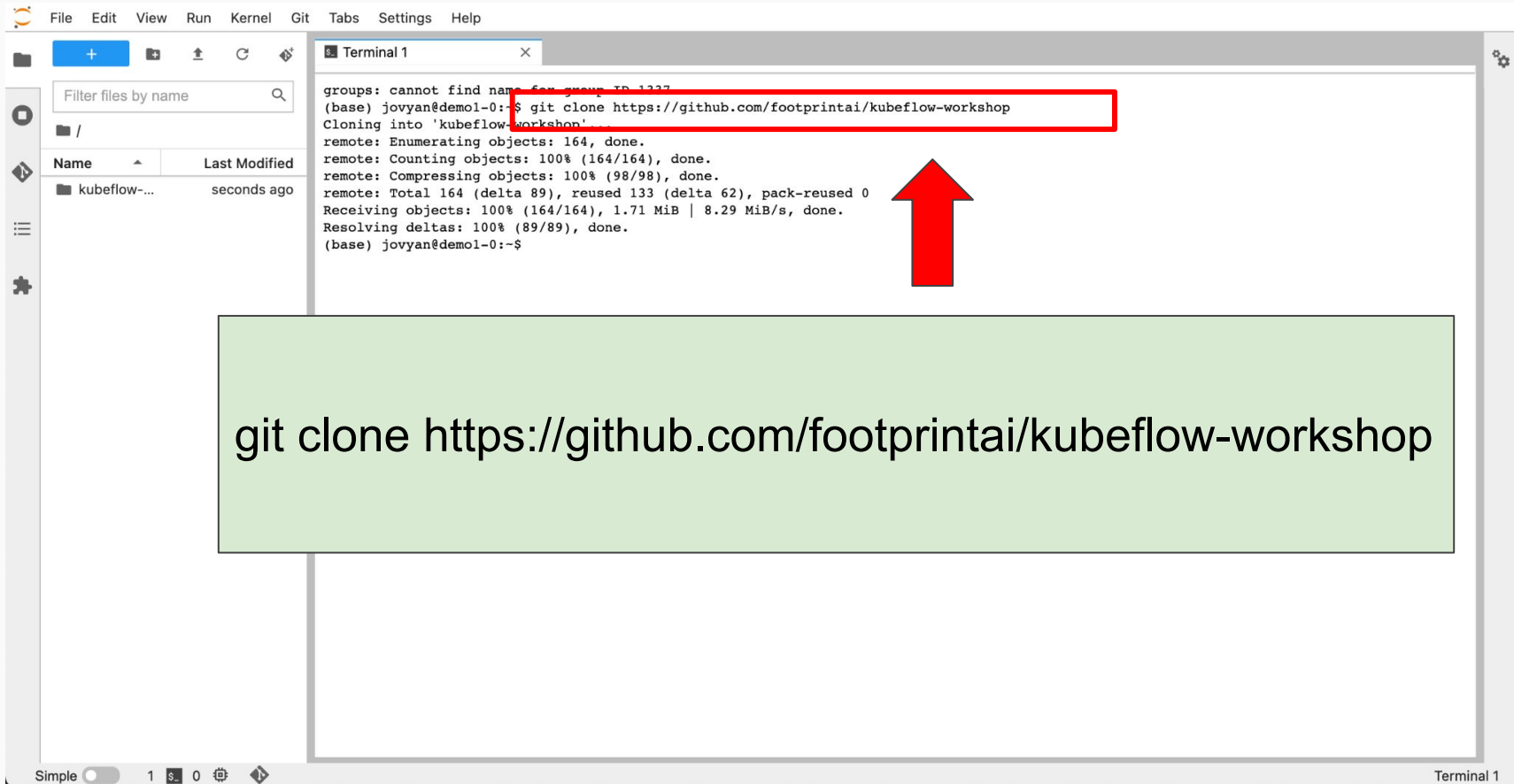
Status	Name	Type	Age	Image	GPUs	CPUs	Memory	Volumes	
✓	demo1		20 hours ago	jupyter-scipy:v1.4	0	0.5	1Gi	⋮	CONNECT  
✓	demo2		2 hours ago	jupyter-tensorflow-full:v1.4	0	0.5	1Gi		CONNECT  



Step2: Use terminal to download the materials (1/3)



Step2: Use terminal to download the materials (2/3)

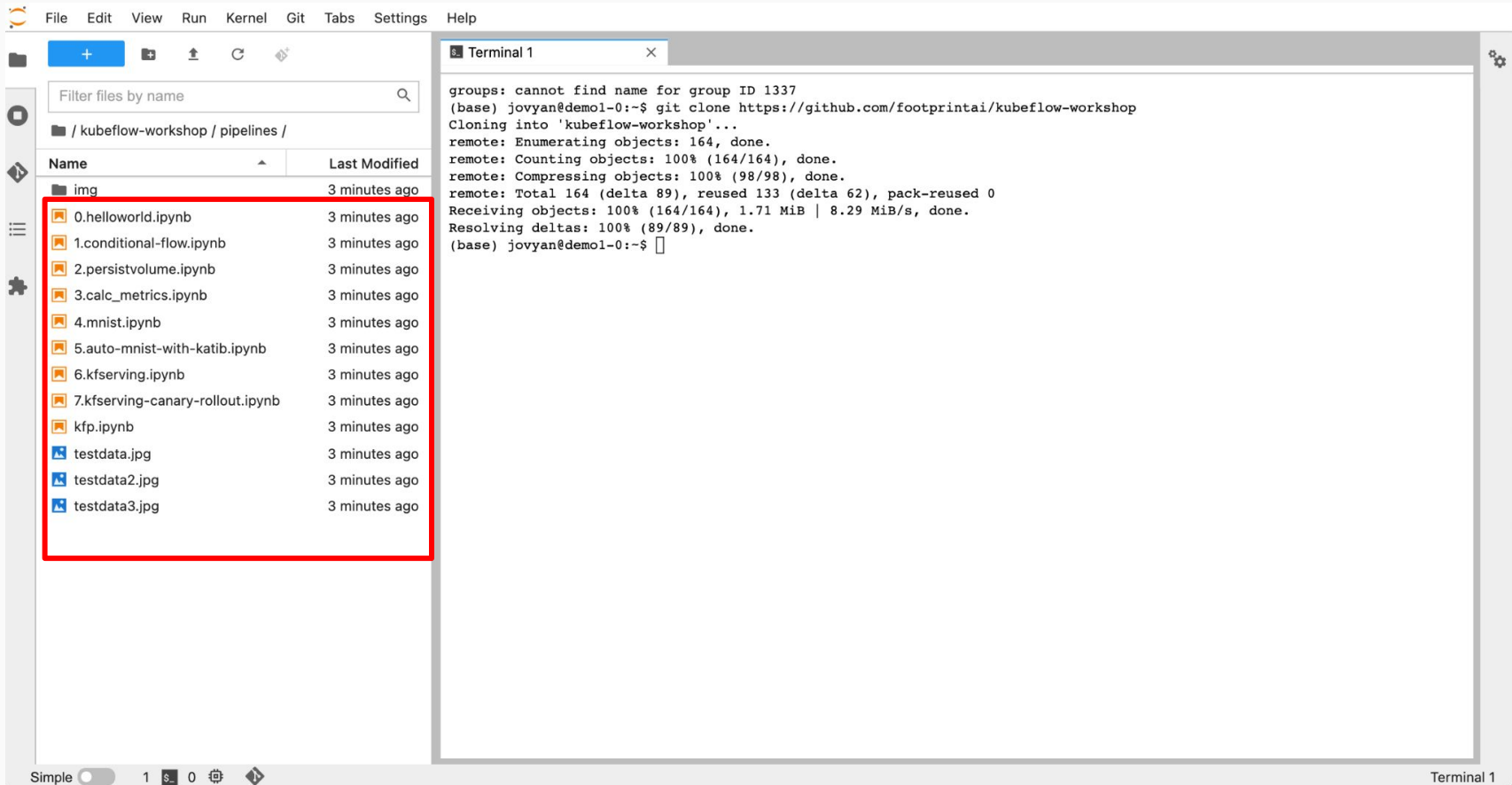


The screenshot shows a JupyterLab environment with a terminal window titled "Terminal 1". The terminal displays the output of a `git clone` command. A red rectangular box highlights the command `(base) jovyan@demo1-0:~$ git clone https://github.com/footprintai/kubeflow-workshop`. A large red arrow points upwards from a green box at the bottom of the image, which contains the same command text: `git clone https://github.com/footprintai/kubeflow-workshop`. The terminal output shows the progress of cloning the repository, including object counting and downloading.

```
groups: cannot find name for group ID 1337
(base) jovyan@demo1-0:~$ git clone https://github.com/footprintai/kubeflow-workshop
Cloning into 'kubeflow-workshop'...
remote: Enumerating objects: 164, done.
remote: Counting objects: 100% (164/164), done.
remote: Compressing objects: 100% (98/98), done.
remote: Total 164 (delta 89), reused 133 (delta 62), pack-reused 0
Receiving objects: 100% (164/164), 1.71 MiB | 8.29 MiB/s, done.
Resolving deltas: 100% (89/89), done.
(base) jovyan@demo1-0:~$
```

git clone https://github.com/footprintai/kubeflow-workshop

Step2: Use terminal to download the materials (3/3)



The screenshot shows a JupyterLab environment. On the left, the file browser displays the directory structure of a cloned repository. The 'img' directory is expanded, showing a list of files. A red box highlights the first 10 files, which are Jupyter Notebook files (.ipynb) and image files (.jpg). The terminal on the right shows the output of a git clone command, indicating that the repository has been successfully cloned.

File Browser:

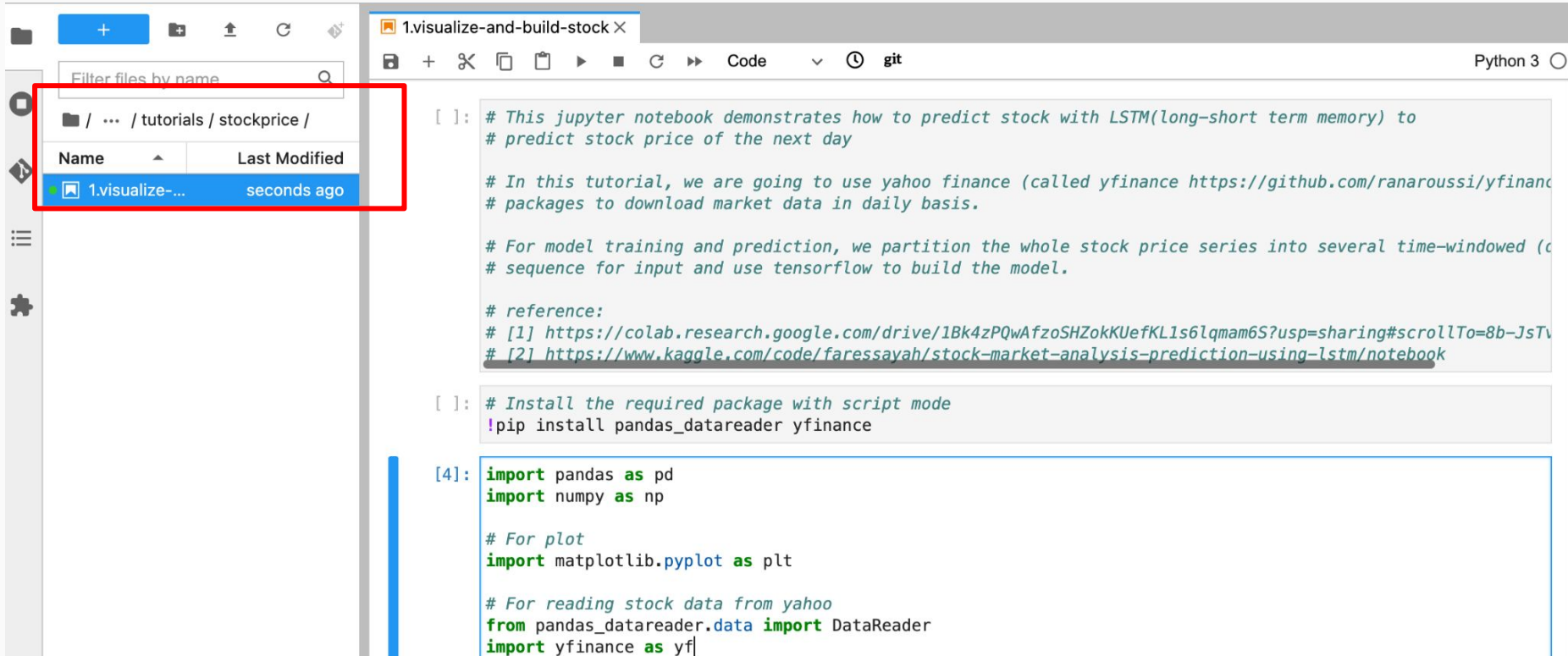
Name	Last Modified
img	3 minutes ago
0.helloworld.ipynb	3 minutes ago
1.conditional-flow.ipynb	3 minutes ago
2.persistvolume.ipynb	3 minutes ago
3.calc_metrics.ipynb	3 minutes ago
4.mnist.ipynb	3 minutes ago
5.auto-mnist-with-katib.ipynb	3 minutes ago
6.kfserving.ipynb	3 minutes ago
7.kfserving-canary-rollout.ipynb	3 minutes ago
kfp.ipynb	3 minutes ago
testdata.jpg	3 minutes ago
testdata2.jpg	3 minutes ago
testdata3.jpg	3 minutes ago

Terminal 1:

```
groups: cannot find name for group ID 1337
(base) jovyan@demol-0:~$ git clone https://github.com/footprintai/kubeflow-workshop
Cloning into 'kubeflow-workshop'...
remote: Enumerating objects: 164, done.
remote: Counting objects: 100% (164/164), done.
remote: Compressing objects: 100% (98/98), done.
remote: Total 164 (delta 89), reused 133 (delta 62), pack-reused 0
Receiving objects: 100% (164/164), 1.71 MiB | 8.29 MiB/s, done.
Resolving deltas: 100% (89/89), done.
(base) jovyan@demol-0:~$
```

Financial Model Builder

Step3: Open financial model ipynb under tutorials/stockprice



The screenshot displays the JupyterLab interface. On the left, the file browser shows the directory structure: / ... / tutorials / stockprice /. A file named 1.visualize-and-build-stock.ipynb is highlighted, with a red box around it. The file's last modified time is 'seconds ago'. On the right, the code editor shows the following Python code:

```
[ ]: # This jupyter notebook demonstrates how to predict stock with LSTM(long-short term memory) to
# predict stock price of the next day

# In this tutorial, we are going to use yahoo finance (called yfinance https://github.com/ranaroussi/yfinance)
# packages to download market data in daily basis.

# For model training and prediction, we partition the whole stock price series into several time-windowed (c
# sequence for input and use tensorflow to build the model.

# reference:
# [1] https://colab.research.google.com/drive/1Bk4zPQwAfzoSHZokKUEfKL1s6lqmam6S?usp=sharing#scrollTo=8b-JsTv
# [2] https://www.kaggle.com/code/faressayah/stock-market-analysis-prediction-using-lstm/notebook

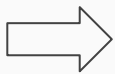
[ ]: # Install the required package with script mode
!pip install pandas_datareader yfinance

[4]: import pandas as pd
import numpy as np

# For plot
import matplotlib.pyplot as plt

# For reading stock data from yahoo
from pandas_datareader.data import DataReader
import yfinance as yf
```

Kubeflow Terms



```

1 | with session_context(scn) as scn as
2 |   (scn := (scn || 'A,B,C'))
3 |
4 | --(scn := (scn || 'A,B,C,D'))
5 |
6 | Insert 179
7 | File 179 Insert 61
8 |
9 | set echo off
10 | set echo on
11 | set serveroutput on
12 | set linesize 100
13 | set trimspool on
14 | set trimspool on
15 | set trimspool on
16 | set trimspool on
17 | set trimspool on
18 | set trimspool on
19 | set trimspool on
20 | set trimspool on
21 | set trimspool on
22 | set trimspool on
23 | set trimspool on
24 | set trimspool on
25 | set trimspool on
26 | set trimspool on
27 | set trimspool on
28 | set trimspool on
29 | set trimspool on
30 | set trimspool on
31 | set trimspool on
32 | set trimspool on
33 | set trimspool on
34 | set trimspool on
35 | set trimspool on
36 | set trimspool on
37 | set trimspool on
38 | set trimspool on
39 | set trimspool on
40 | set trimspool on
41 | set trimspool on
42 | set trimspool on
43 | set trimspool on
44 | set trimspool on
45 | set trimspool on
46 | set trimspool on
47 | set trimspool on
48 | set trimspool on
49 | set trimspool on
50 | set trimspool on
51 | set trimspool on
52 | set trimspool on
53 | set trimspool on
54 | set trimspool on
55 | set trimspool on
56 | set trimspool on
57 | set trimspool on
58 | set trimspool on
59 | set trimspool on
60 | set trimspool on
61 | set trimspool on
62 | set trimspool on
63 | set trimspool on
64 | set trimspool on
65 | set trimspool on
66 | set trimspool on
67 | set trimspool on
68 | set trimspool on
69 | set trimspool on
70 | set trimspool on
71 | set trimspool on
72 | set trimspool on
73 | set trimspool on
74 | set trimspool on
75 | set trimspool on
76 | set trimspool on
77 | set trimspool on
78 | set trimspool on
79 | set trimspool on
80 | set trimspool on
81 | set trimspool on
82 | set trimspool on
83 | set trimspool on
84 | set trimspool on
85 | set trimspool on
86 | set trimspool on
87 | set trimspool on
88 | set trimspool on
89 | set trimspool on
90 | set trimspool on
91 | set trimspool on
92 | set trimspool on
93 | set trimspool on
94 | set trimspool on
95 | set trimspool on
96 | set trimspool on
97 | set trimspool on
98 | set trimspool on
99 | set trimspool on
100 | set trimspool on

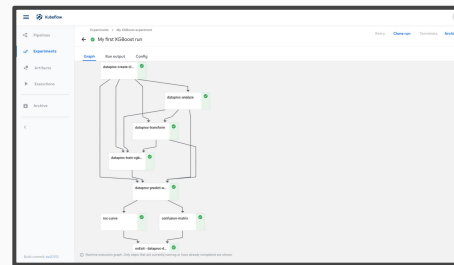
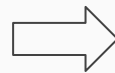
```

Pipeline Code

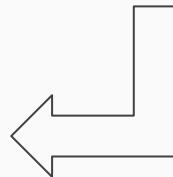
Compiled

Workflow Resource

Create a Pipeline



Create Run



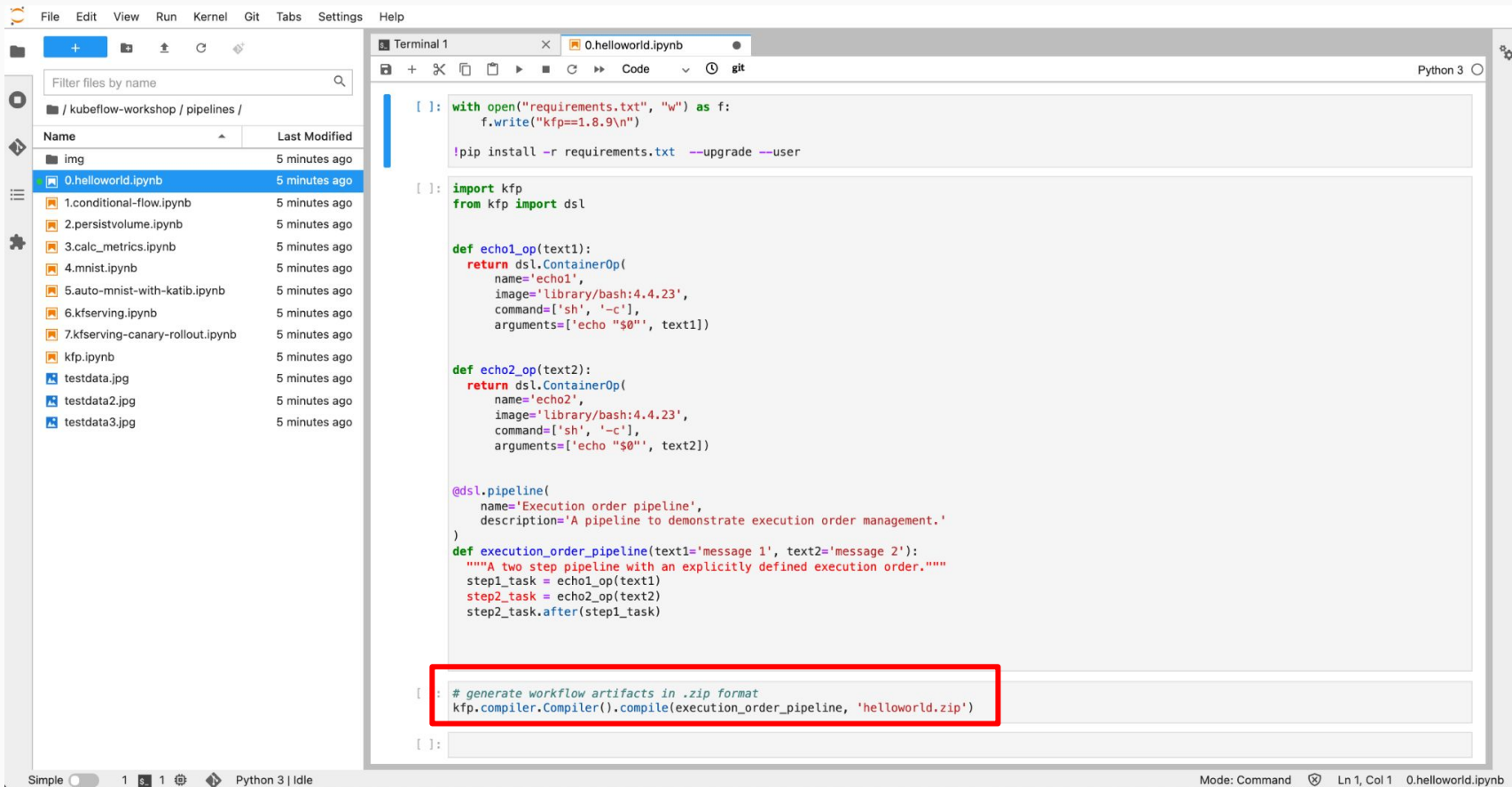
Experiment1

Experiment2

Runs

Hello World Example

Step4: Compile helloworld.ipynb (1/2)

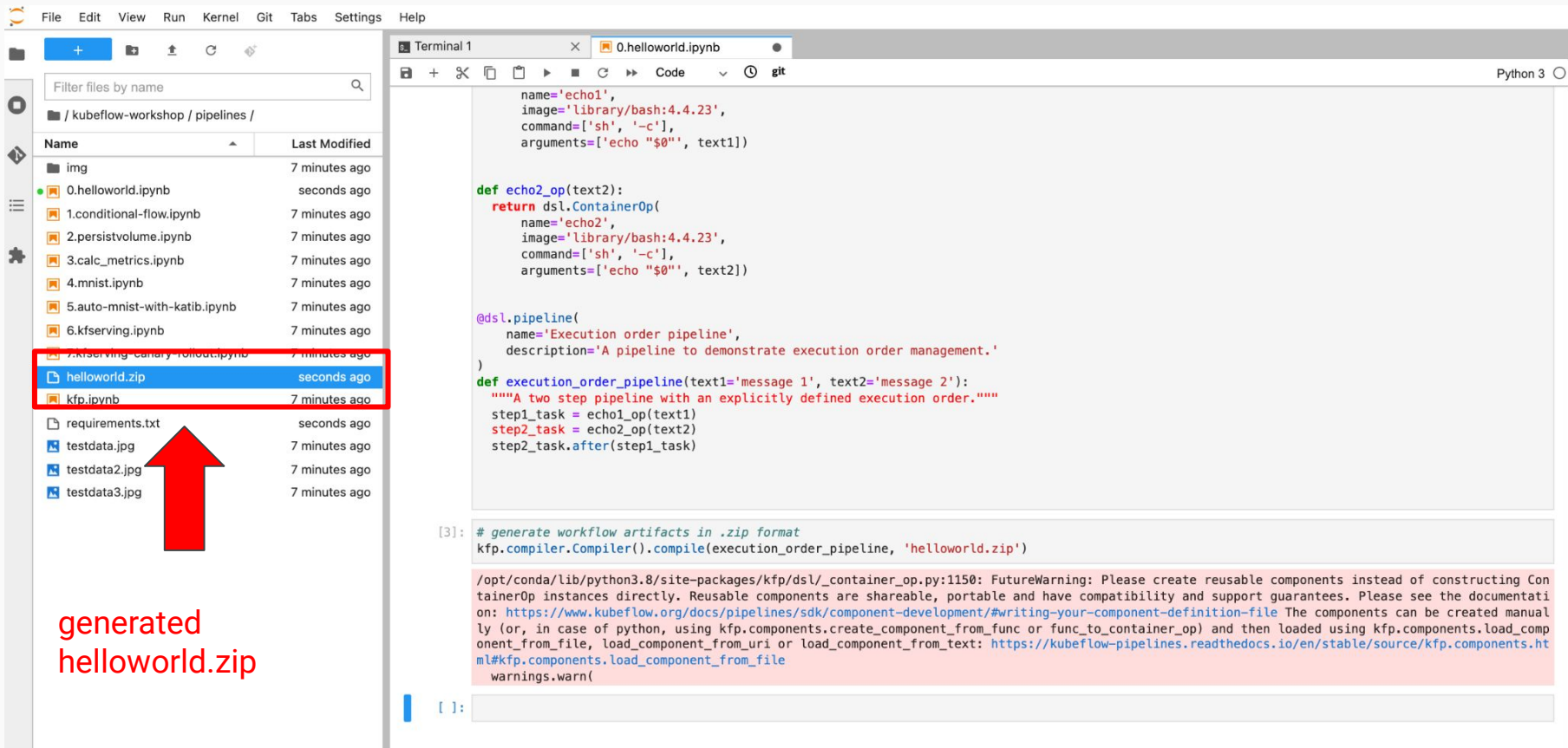


The screenshot displays the JupyterLab environment. On the left, a file browser shows the directory structure: `/ kubeflow-workshop / pipelines /`. A list of files is shown, including `img`, `0.helloworld.ipynb` (selected), `1.conditional-flow.ipynb`, `2.persistvolume.ipynb`, `3.calc_metrics.ipynb`, `4.mnist.ipynb`, `5.auto-mnist-with-katib.ipynb`, `6.kfserving.ipynb`, `7.kfserving-canary-rollout.ipynb`, `kfp.ipynb`, `testdata.jpg`, `testdata2.jpg`, and `testdata3.jpg`. The main editor window shows the code for `0.helloworld.ipynb`. The code defines two tasks, `echo1_op` and `echo2_op`, which return `dsl.ContainerOp` objects. It also defines a pipeline, `execution_order_pipeline`, which uses these tasks. The final line of code, which generates workflow artifacts in .zip format, is highlighted with a red box:

```
[ ]: # generate workflow artifacts in .zip format
kfp.compiler.Compiler().compile(execution_order_pipeline, 'helloworld.zip')
```

The bottom status bar indicates the current mode is 'Command', the file is 'Ln 1, Col 1', and the notebook is '0.helloworld.ipynb'.

Step4: Compile helloworld.ipynb (2/2)



File Edit View Run Kernel Git Tabs Settings Help

Filter files by name

/ kubeflow-workshop / pipelines /

Name	Last Modified
img	7 minutes ago
0.helloworld.ipynb	seconds ago
1.conditional-flow.ipynb	7 minutes ago
2.persistvolume.ipynb	7 minutes ago
3.calc_metrics.ipynb	7 minutes ago
4.mnist.ipynb	7 minutes ago
5.auto-mnist-with-katib.ipynb	7 minutes ago
6.kfserving.ipynb	7 minutes ago
7.kfserving-canary-rollout.ipynb	7 minutes ago
helloworld.zip	seconds ago
kfp.ipynb	7 minutes ago
requirements.txt	seconds ago
testdata.jpg	7 minutes ago
testdata2.jpg	7 minutes ago
testdata3.jpg	7 minutes ago

generated helloworld.zip

Terminal 1 0.helloworld.ipynb Python 3

```
name='echo1',
image='library/bash:4.4.23',
command=['sh', '-c'],
arguments=['echo "$0"', text1])

def echo2_op(text2):
    return dsl.ContainerOp(
        name='echo2',
        image='library/bash:4.4.23',
        command=['sh', '-c'],
        arguments=['echo "$0"', text2])


@dsl.pipeline(
    name='Execution order pipeline',
    description='A pipeline to demonstrate execution order management.'
)

def execution_order_pipeline(text1='message 1', text2='message 2'):
    """A two step pipeline with an explicitly defined execution order."""
    step1_task = echo1_op(text1)
    step2_task = echo2_op(text2)
    step2_task.after(step1_task)

[3]: # generate workflow artifacts in .zip format
kfp.compiler.Compiler().compile(execution_order_pipeline, 'helloworld.zip')

/opt/conda/lib/python3.8/site-packages/kfp/dsl/_container_op.py:1150: FutureWarning: Please create reusable components instead of constructing ContainerOp instances directly. Reusable components are shareable, portable and have compatibility and support guarantees. Please see the documentation on: https://www.kubeflow.org/docs/pipelines/sdk/component-development/#writing-your-component-definition-file The components can be created manually (or, in case of python, using kfp.components.create_component_from_func or func_to_container_op) and then loaded using kfp.components.load_component_from_file, load_component_from_uri or load_component_from_text: https://kubeflow-pipelines.readthedocs.io/en/stable/source/kfp.components.html#kfp.components.load\_component\_from\_file
warnings.warn(
```

Step4: Create a Pipeline (1/7)

 **Kubeflow**

Home

Notebooks

Tensorboards

Volumes

Models

Experiments (AutoML)

Experiments (KFP)

Pipelines

Runs

Recurring Runs

Artifacts

Executions

Manage Contributors

kubeflow-user-example-c... ▾

↗

+ Upload pipeline

Refresh

Delete

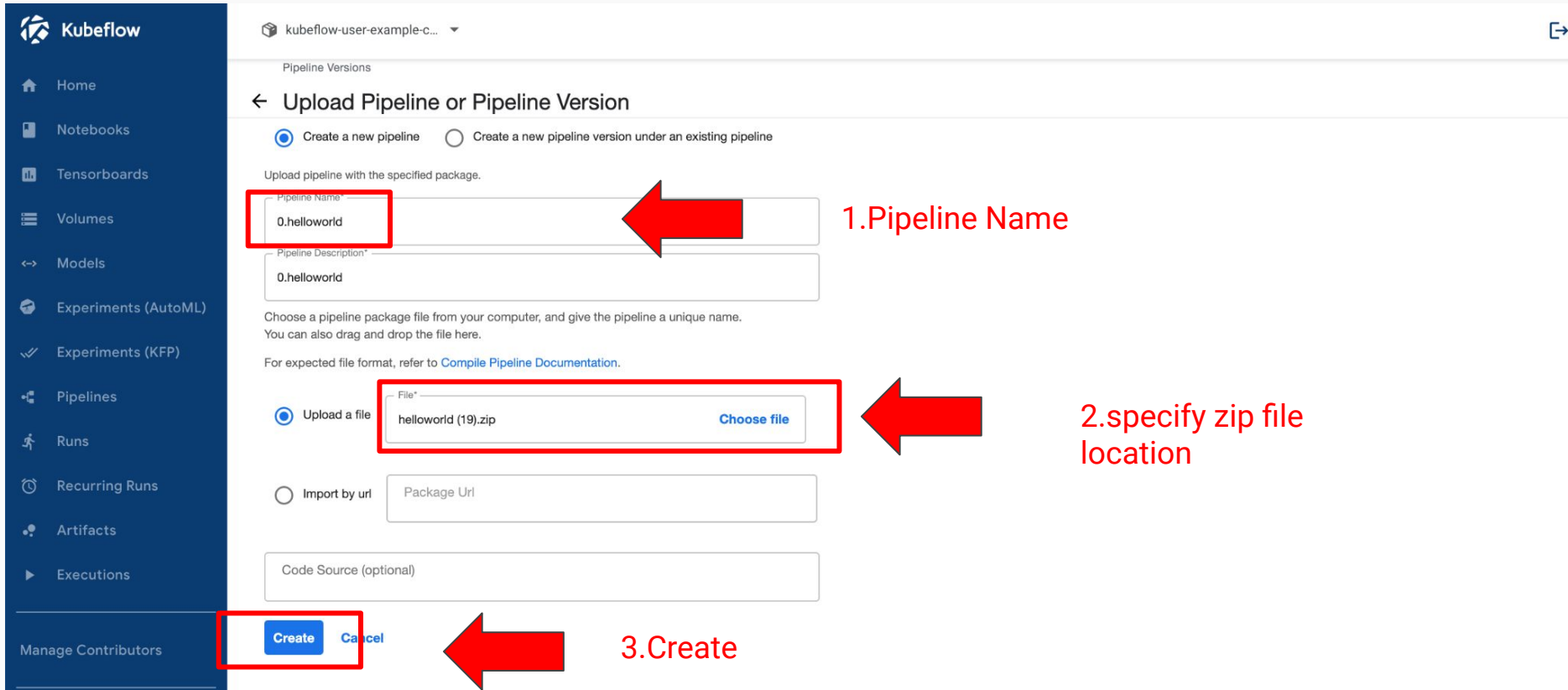
Filter pipelines

▾

<input type="checkbox"/>	Pipeline name	Description	Upload	↓
<input type="checkbox"/>	▶ [Tutorial] V2 lightweight Python com...	source code Shows different component input and output options for KFP v2 components.	11/30/2021, 1:02:25 PM	
<input type="checkbox"/>	▶ [Tutorial] DSL - Control structures	source code Shows how to use conditional execution and exit handlers. This pipeline will randomly fail to demonstr...	11/30/2021, 1:02:24 PM	
<input type="checkbox"/>	▶ [Tutorial] Data passing in python co...	source code Shows how to pass data between python components.	11/30/2021, 1:02:23 PM	
<input type="checkbox"/>	▶ [Demo] TFX - Taxi tip prediction mod...	source code GCP Permission requirements . Example pipeline that does classification with model analysis based on...	11/30/2021, 1:02:22 PM	
<input type="checkbox"/>	▶ [Demo] XGBoost - Iterative model tra...	source code This sample demonstrates iterative training using a train-eval-check recursive loop. The main pipeline ...	11/30/2021, 1:02:21 PM	

Rows per page: 10 ▾ < >

Step4: Create a Pipeline (2/7)



Kubeflow

kubeflow-user-example-c...

Pipeline Versions

← Upload Pipeline or Pipeline Version

☒ Create a new pipeline ☐ Create a new pipeline version under an existing pipeline

Upload pipeline with the specified package.

Pipeline Name* **0.helloworld** ← 1. Pipeline Name

Pipeline Description* 0.helloworld

Choose a pipeline package file from your computer, and give the pipeline a unique name. You can also drag and drop the file here.

For expected file format, refer to [Compile Pipeline Documentation](#).

☒ Upload a file **File* helloworld (19).zip** Choose file ← 2. specify zip file location

☐ Import by url Package Url

Code Source (optional)

Create Cancel ← 3. Create

Manage Contributors

Step4: Create a Pipeline (3/7)

The screenshot displays the Kubeflow Pipelines web interface. On the left is a dark blue sidebar with navigation links: Home, Notebooks, Tensorboards, Volumes, Models, Experiments (AutoML), Experiments (KFP), Pipelines, Runs, Recurring Runs, Artifacts, and Executions. At the bottom of the sidebar is a link for 'Manage Contributors'. The main content area has a header with the user 'kubeflow-user-example-c...' and a dropdown arrow. Below this, the 'Pipelines' section shows a breadcrumb '← 0.helloworld (0.helloworld)'. To the right of the breadcrumb are three buttons: '+ Create run', '+ Upload version', and '+ Create experiment' (which is highlighted with a red rectangular box). Further right is a 'Delete' link. Below the buttons, there are tabs for 'Graph' and 'YAML', with 'Graph' selected. A 'Simplify Graph' toggle switch is present. The graph itself shows two nodes, 'echo1' and 'echo2', connected by a downward arrow. At the bottom of the main area is a 'Summary' panel with a 'Hide' link. The summary contains the following information: ID: 6f25028f-01e3-4acd-9389-7ec2031fb04b, Version: 0.helloworld (with a dropdown arrow), and Version source.

Kubeflow

kubeflow-user-example-c...

Pipelines

← 0.helloworld (0.helloworld)

+ Create run + Upload version + Create experiment Delete

Graph YAML

Simplify Graph

echo1

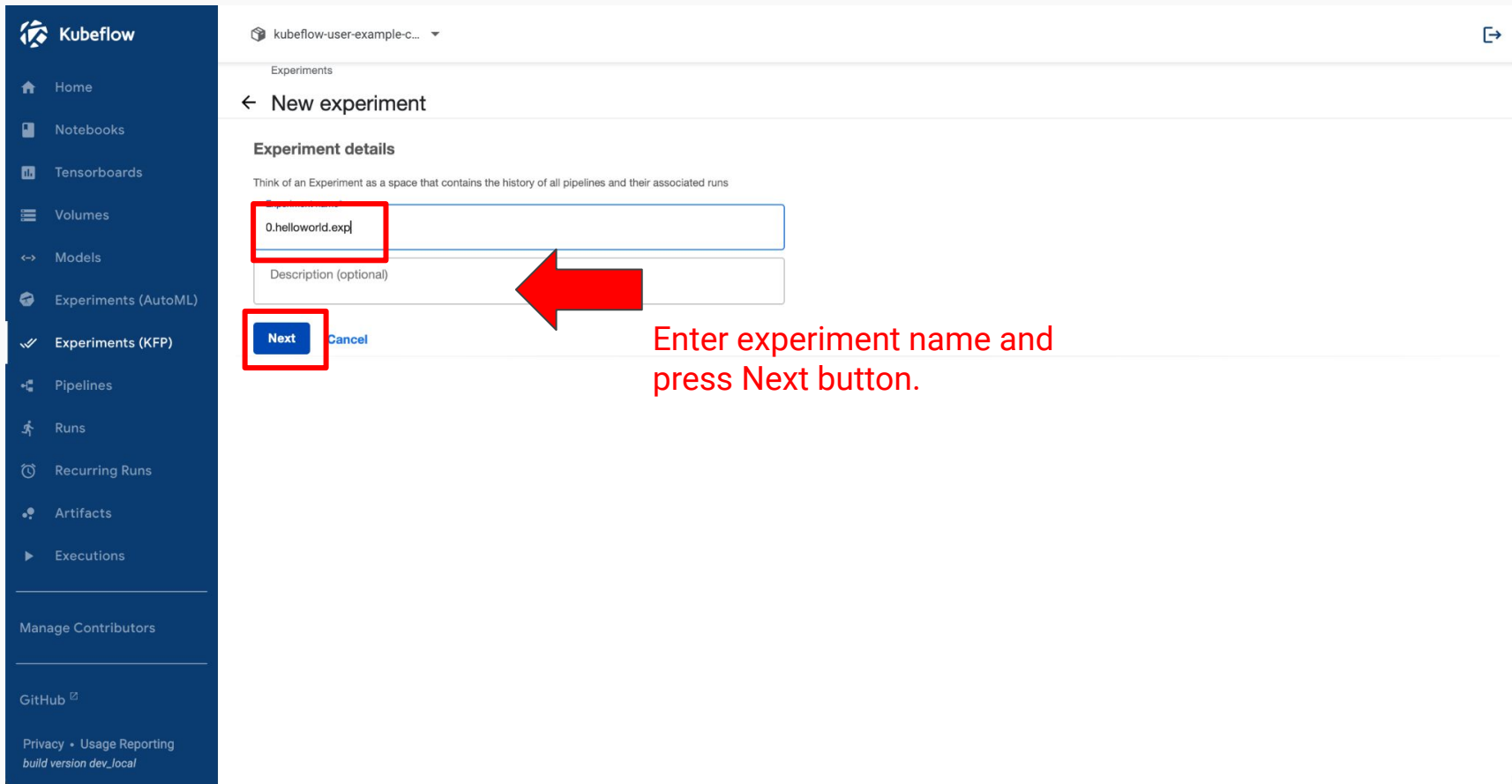
echo2

Summary Hide

ID
6f25028f-01e3-4acd-9389-7ec2031fb04b
Version
0.helloworld
Version source

Create an experiment


Step4: Create a Pipeline (4/7)



The screenshot displays the Kubeflow console interface for creating a new experiment. On the left is a dark blue sidebar with navigation links: Home, Notebooks, Tensorboards, Volumes, Models, Experiments (AutoML), Experiments (KFP) (selected), Pipelines, Runs, Recurring Runs, Artifacts, and Executions. Below these are links for Manage Contributors, GitHub, and footer text: Privacy • Usage Reporting, build version dev_local.

The main content area is titled 'New experiment' and includes a sub-section 'Experiment details' with the instruction: 'Think of an Experiment as a space that contains the history of all pipelines and their associated runs'. There are two input fields: 'Experiment name' containing '0.helloworld.exp' and 'Description (optional)'. Both fields are outlined in blue. A red box highlights the 'Experiment name' field, and a red arrow points from the text 'Enter experiment name and press Next button.' to the 'Next' button. The 'Next' button is a blue rectangle with white text, and the 'Cancel' button is a blue text link.

Step4: Create a Pipeline (5/7)

 Kubeflow

Home

Notebooks

Tensorboards

Volumes

Models

Experiments (AutoML)

Experiments (KFP)

Pipelines

Runs

Recurring Runs

Artifacts

Executions

Manage Contributors

GitHub

Documentation

Privacy • Usage Reporting
build version dev_local

kubeflow-user-example-c...

Run details

Pipeline*

0.helloworld

Choose

Pipeline Version*

0.helloworld

Choose

Run name*

Run of 0.helloworld [1e261]

Description (optional)

This run will be associated with the following experiment

Experiment*

0.helloworld.exp

Choose

This run will use the following Kubernetes service account.

Service Account (Optional)

Run Type

☒ One-off

☐ Recurring

Run parameters

Specify parameters required by the pipeline

text1

message 1

text2

message 2

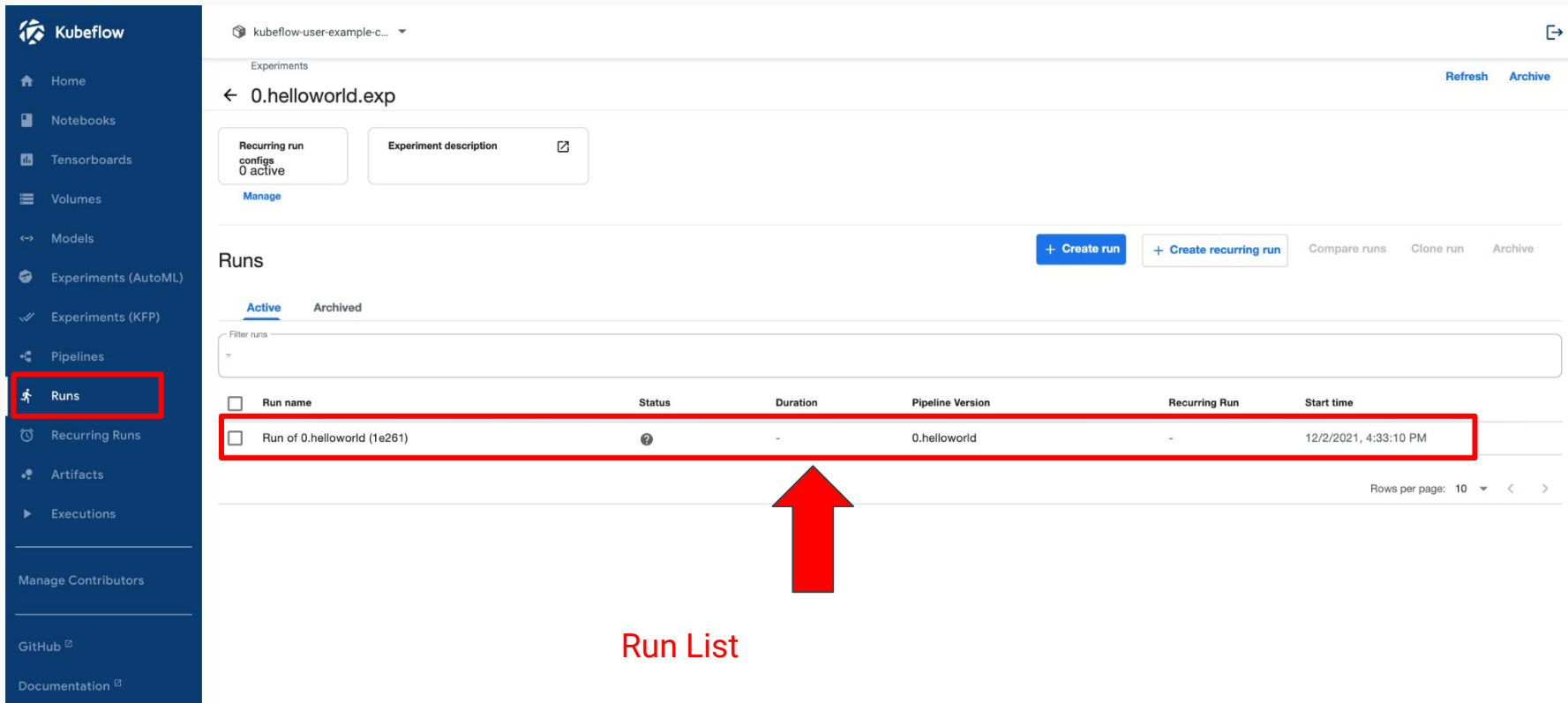
Start

Skip this step

1. Run a pipeline and specify its version

2. Add its experiment name

Step4: Create a Pipeline (6/7)



Kubeflow

Home

Notebooks

Tensorboards

Volumes

Models

Experiments (AutoML)

Experiments (KFP)

Pipelines

Runs

Recurring Runs

Artifacts

Executions

Manage Contributors

GitHub

Documentation

kubeflow-user-example-c...

Experiments

Refresh Archive

← 0.helloworld.exp

Recurring run configs 0 active Manage

Experiment description

Runs

+ Create run + Create recurring run Compare runs Clone run Archive

Active Archived

Filter runs

<input type="checkbox"/>	Run name	Status	Duration	Pipeline Version	Recurring Run	Start time
<input type="checkbox"/>	Run of 0.helloworld (1e261)	?	-	0.helloworld	-	12/2/2021, 4:33:10 PM

Rows per page: 10 < >

Run List

Step4: Create a Pipeline (7/7)

The screenshot displays the Kubeflow dashboard interface. On the left is a dark blue sidebar with navigation links: Home, Notebooks, Tensorboards, Volumes, Models, Experiments (AutoML), Experiments (KFP), Pipelines, Runs, Recurring Runs, Artifacts, Executions, Manage Contributors, and GitHub. The main content area shows the 'Run of 0.helloworld (1e261)' page. At the top, there are tabs for 'Graph', 'Run output', and 'Config'. The 'Graph' tab is active, showing a pipeline graph with two nodes, 'echo1' and 'echo2', connected by a downward arrow. Both nodes have a green checkmark and a refresh icon. A red rectangle highlights this graph. To the right of the graph is a modal window titled 'execution-order-pipeline-fxxfn-4223123588'. It has tabs for 'Input/Output', 'Visualizations', 'Details', 'Volumes', 'Logs', 'Pod', 'Events', and 'ML Metadata'. The 'Input/Output' tab is selected. It lists 'Input parameters' (text1: message 1), 'Input artifacts', 'Output parameters', and 'Output artifacts'. The 'Output artifacts' section is highlighted with a red rectangle and contains 'main-logs' with a link to 'minio://mlpipeline/artifacts/execution-order-pipeline-qvlt5/2021/11/30/execution-order-pipeline-qvlt5-137025384/main.log'. A large red arrow points from the text '運行結果輸出' (Output of execution results) to the 'main-logs' artifact link.

Kubeflow

Home
Notebooks
Tensorboards
Volumes
Models
Experiments (AutoML)
Experiments (KFP)
Pipelines
Runs
Recurring Runs
Artifacts
Executions
Manage Contributors
GitHub

kubeflow-user-example-c...

Experiments > 0.helloworld.exp

← Run of 0.helloworld (1e261)

Graph Run output Config

Simplify Graph

echo1

echo2

execution-order-pipeline-fxxfn-4223123588

Input/Output Visualizations Details Volumes Logs Pod Events ML Metadata

Input parameters

text1 message 1

Input artifacts

Output parameters

Output artifacts

main-logs minio://mlpipeline/artifacts/execution-order-pipeline-qvlt5/2021/11/30/execution-order-pipeline-qvlt5-137025384/main.log View All

message 1

運行結果輸出

An aerial photograph of the New York City skyline at dusk. The sky is a mix of dark blue and orange, with scattered clouds. The city is densely packed with skyscrapers, many of which are illuminated with their interior lights. The Empire State Building is prominent in the center, with its top lit in red and green. The Hudson River is visible in the background, with the New York-New York Hotel & Casino's replica of the Empire State Building on the right side of the image.

Thank You
Questions?

Contact Us:
partnership@footprint-ai.com