

# 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

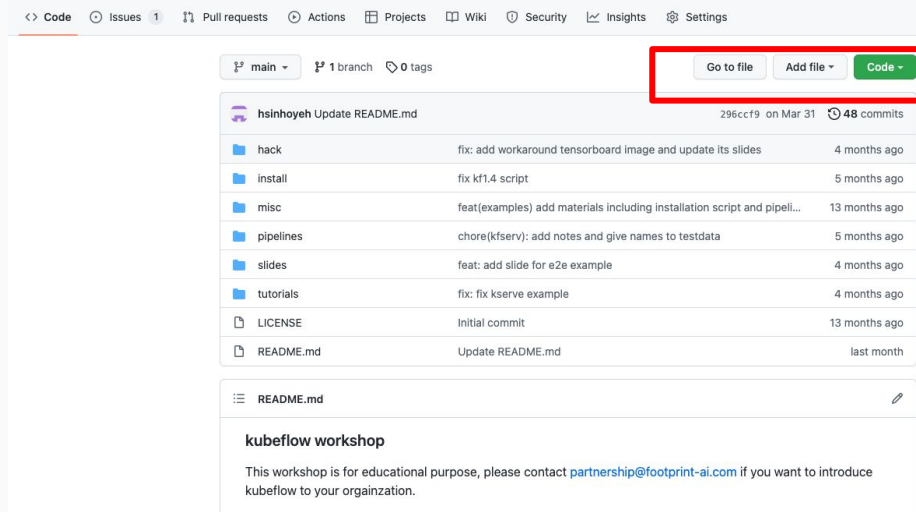
<https://reurl.cc/yrkGrE>



# Materials

git clone

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



The screenshot shows the GitHub repository page for `FootprintAI/kubeflow-workshop`. The repository is in 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
```

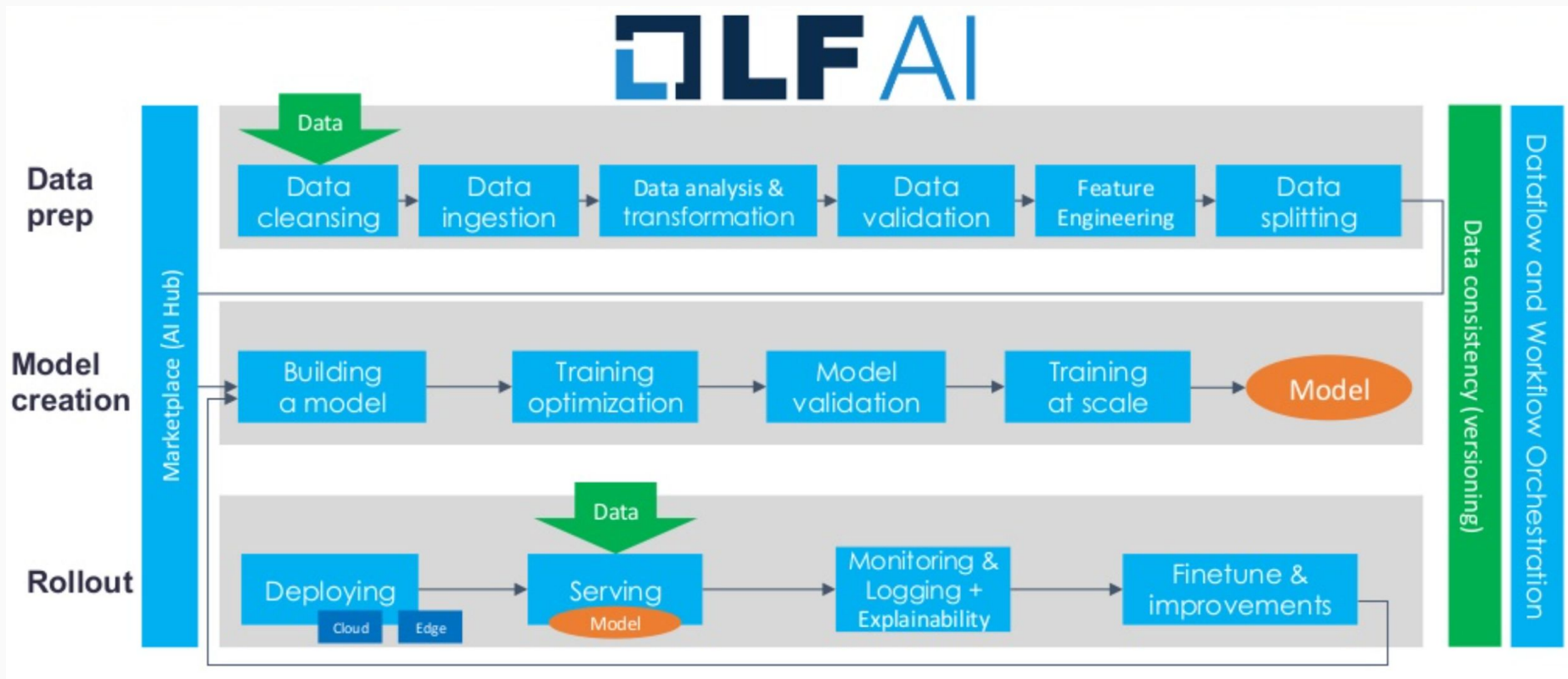
This workshop is for educational purpose, please contact [partnership@footprint-ai.com](mailto:partnership@footprint-ai.com) if you want to introduce kubeflow to your organization.

# 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

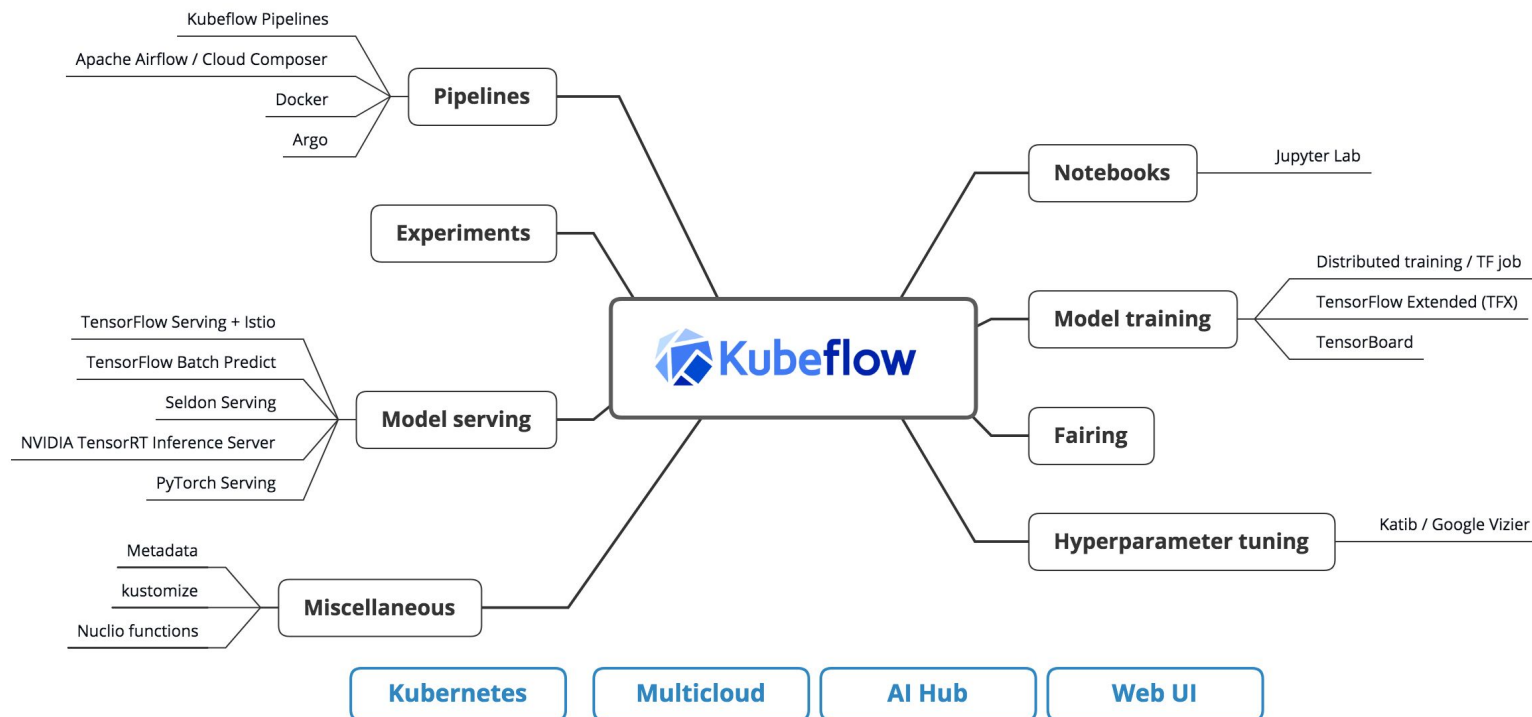


# Real-world Machine Learning Application - End-to-End ML LifeCycle



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	?

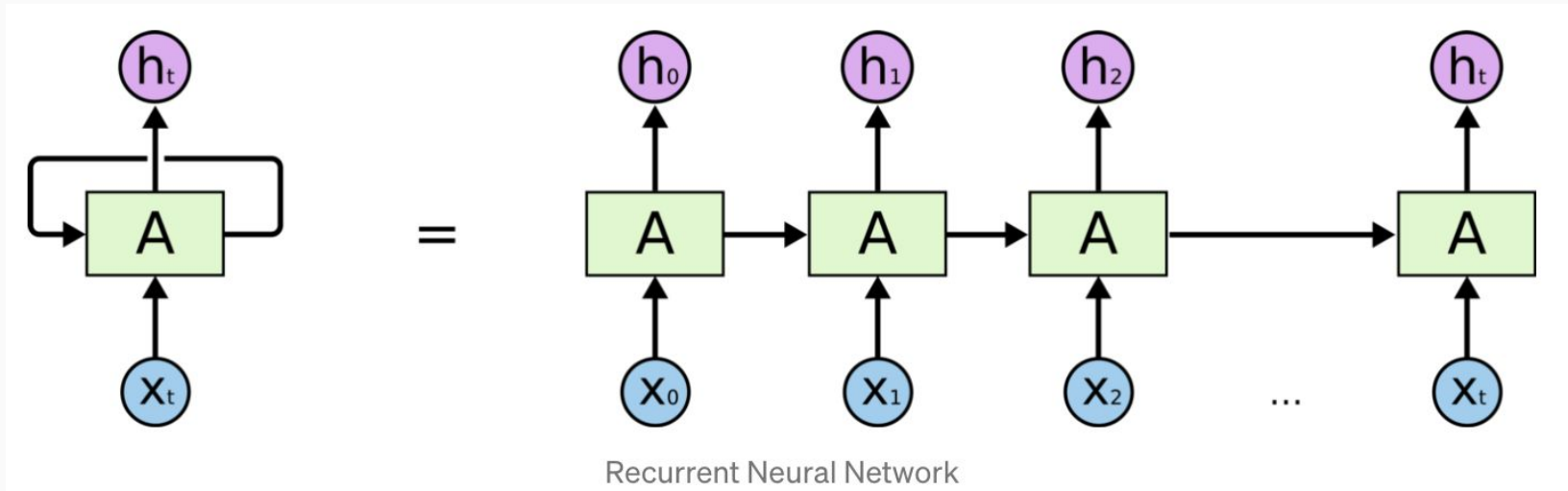
## What sequences means in a sentence?

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.

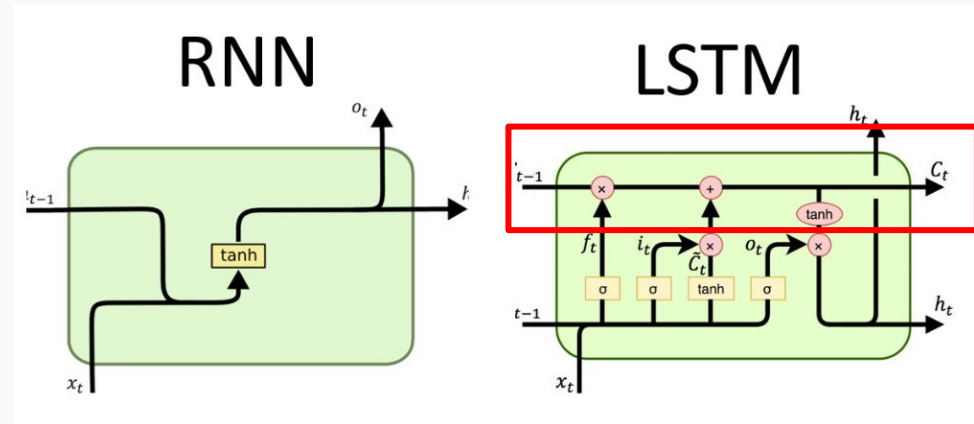


# What Is LSTM (Long-short term memory)?

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



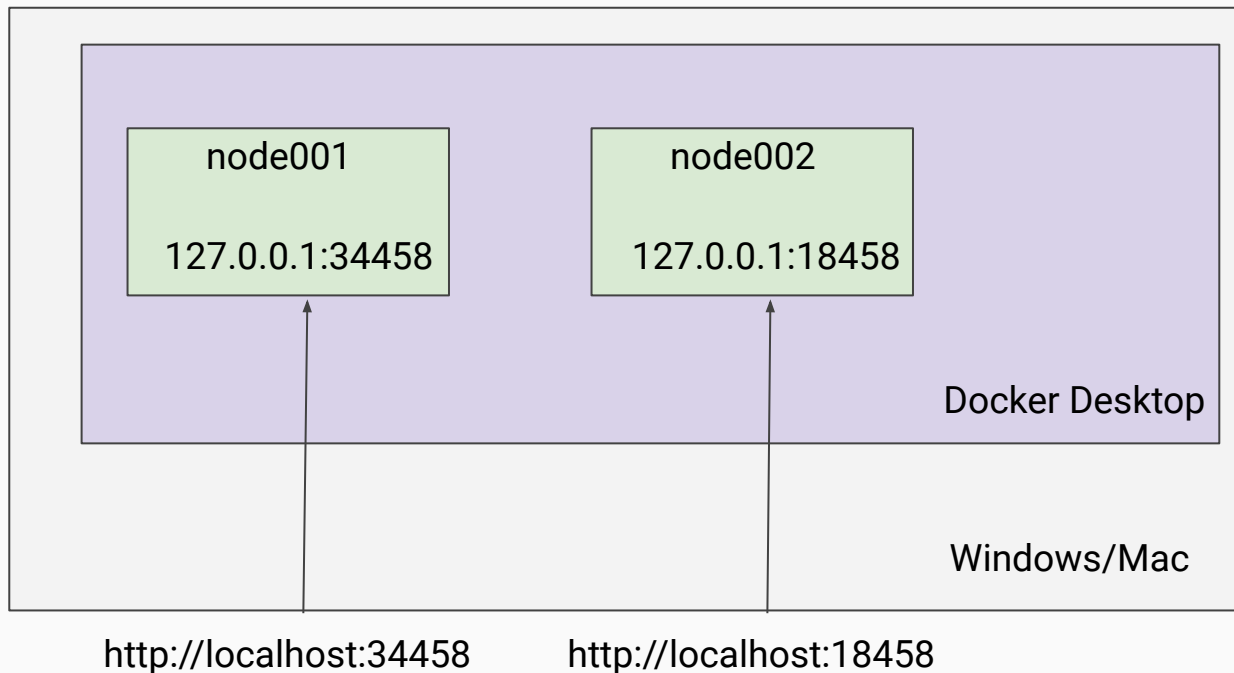
Windows size 3, we look at last three days data

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: few steps setup

```
// install dockerd
wget https://raw.githubusercontent.com/FootprintAI/multikf/main/docker/linux.sh
chmod +x linux.sh
sudo ./linux.sh
```

```
// install multikf
wget https://github.com/FootprintAI/multikf/raw/main/build/multikf.linux
chmod +x multikf.linux
```

```
// add an instances with port 80/443 exported
./multikf.linux add node002 --export_ports 80:80,443:443
```

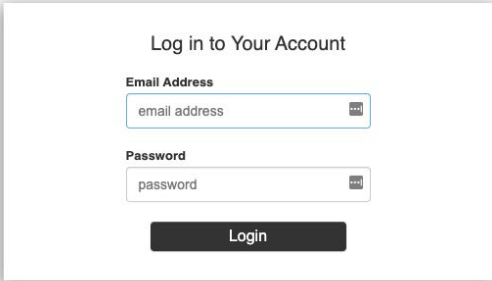
```
// connect kubeflow
./multikf.linux connect kubeflow node002
```



Wait! 所以我說那個帳號密碼呢?

Account: user@example.com

Password: 12341234



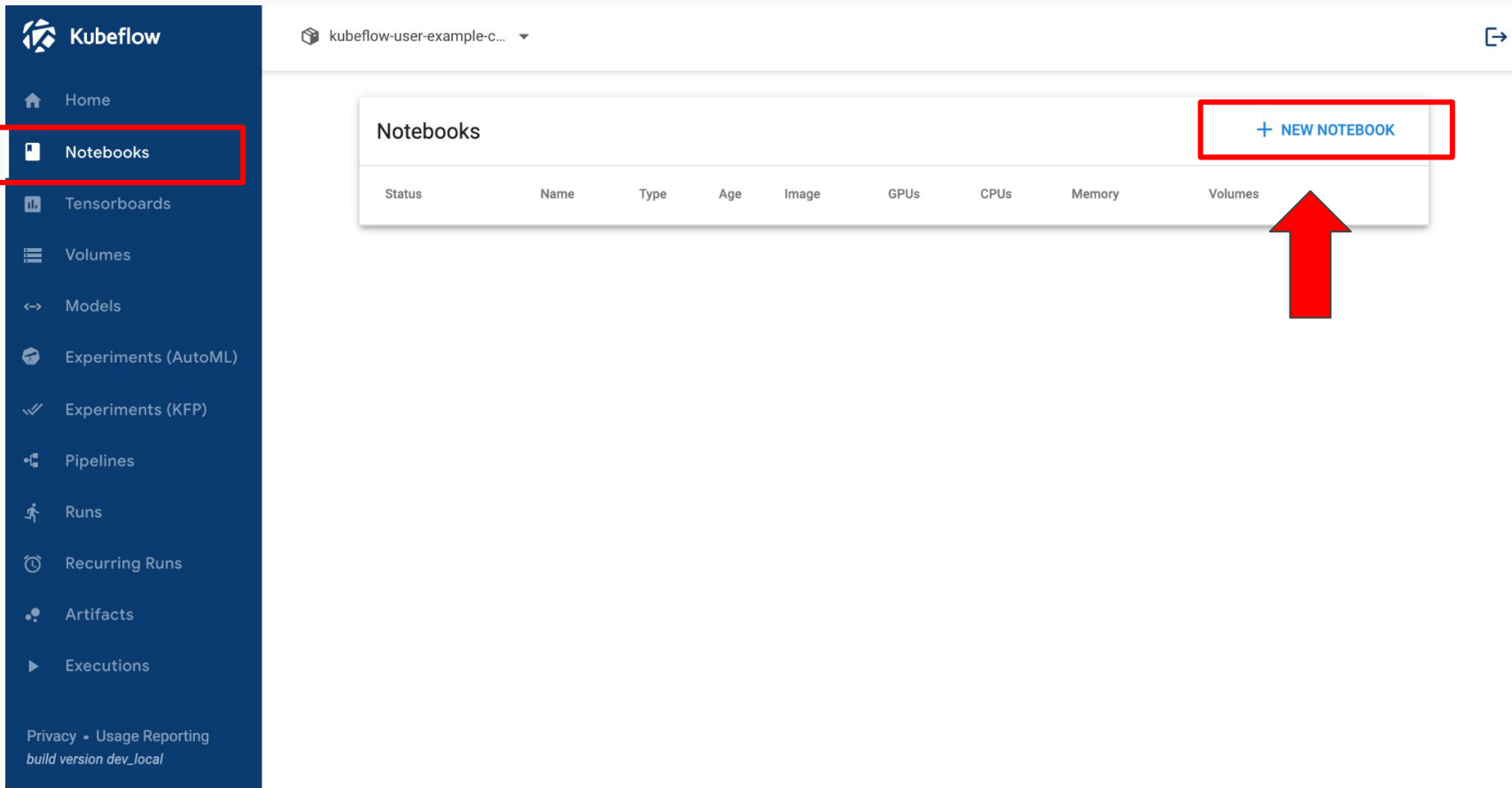
Log in to Your Account

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 of columns: Status, Name, Type, Age, Image, GPUs, CPUs, Memory, and Volumes. A red rectangle highlights the '+ NEW NOTEBOOK' button in the top right corner of the Notebooks section. A large red arrow points upwards towards the 'Volumes' column header in the table.

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](#)

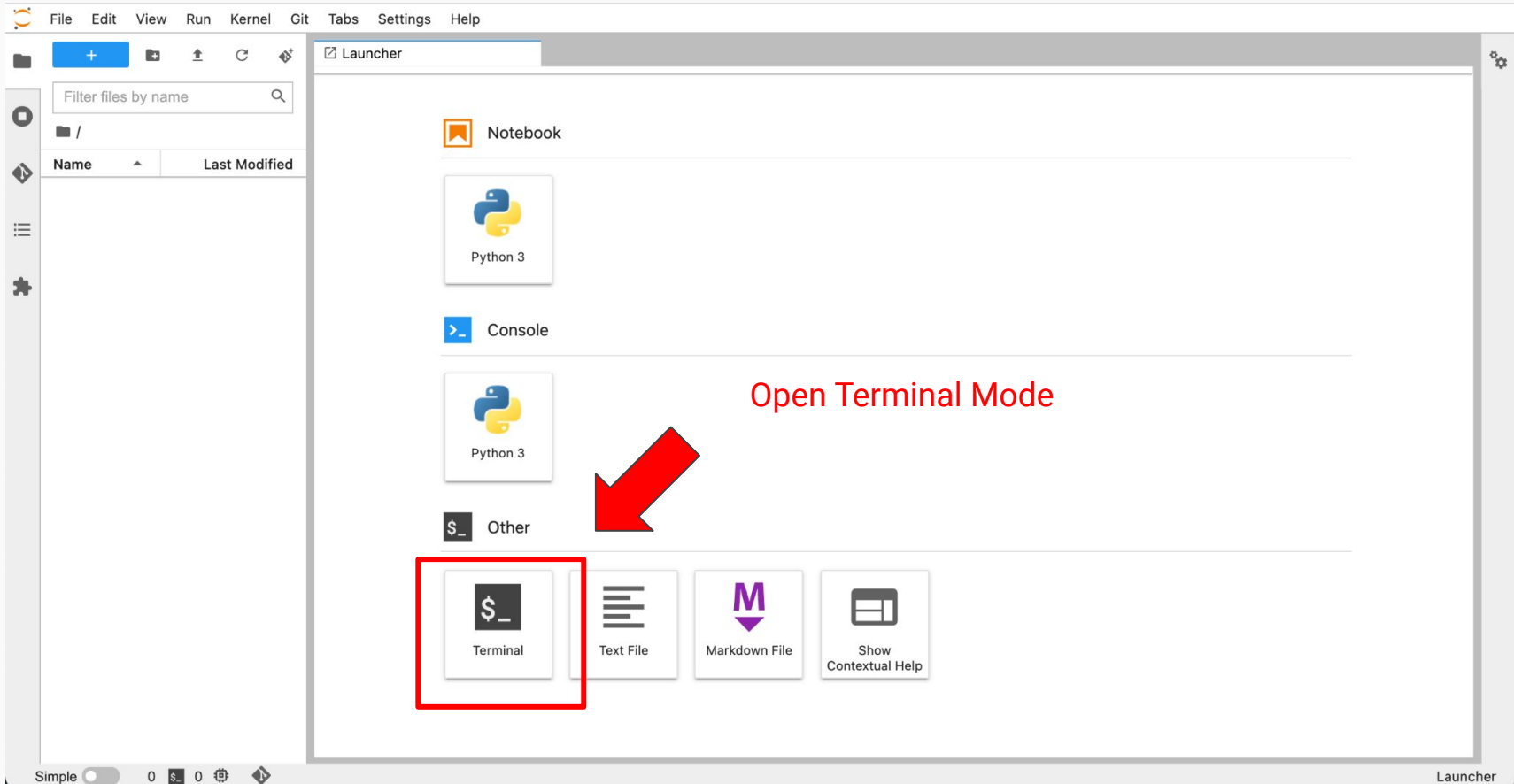
kubeflow-user-example-c... 

Notebooks [+ NEW NOTEBOOK](#)

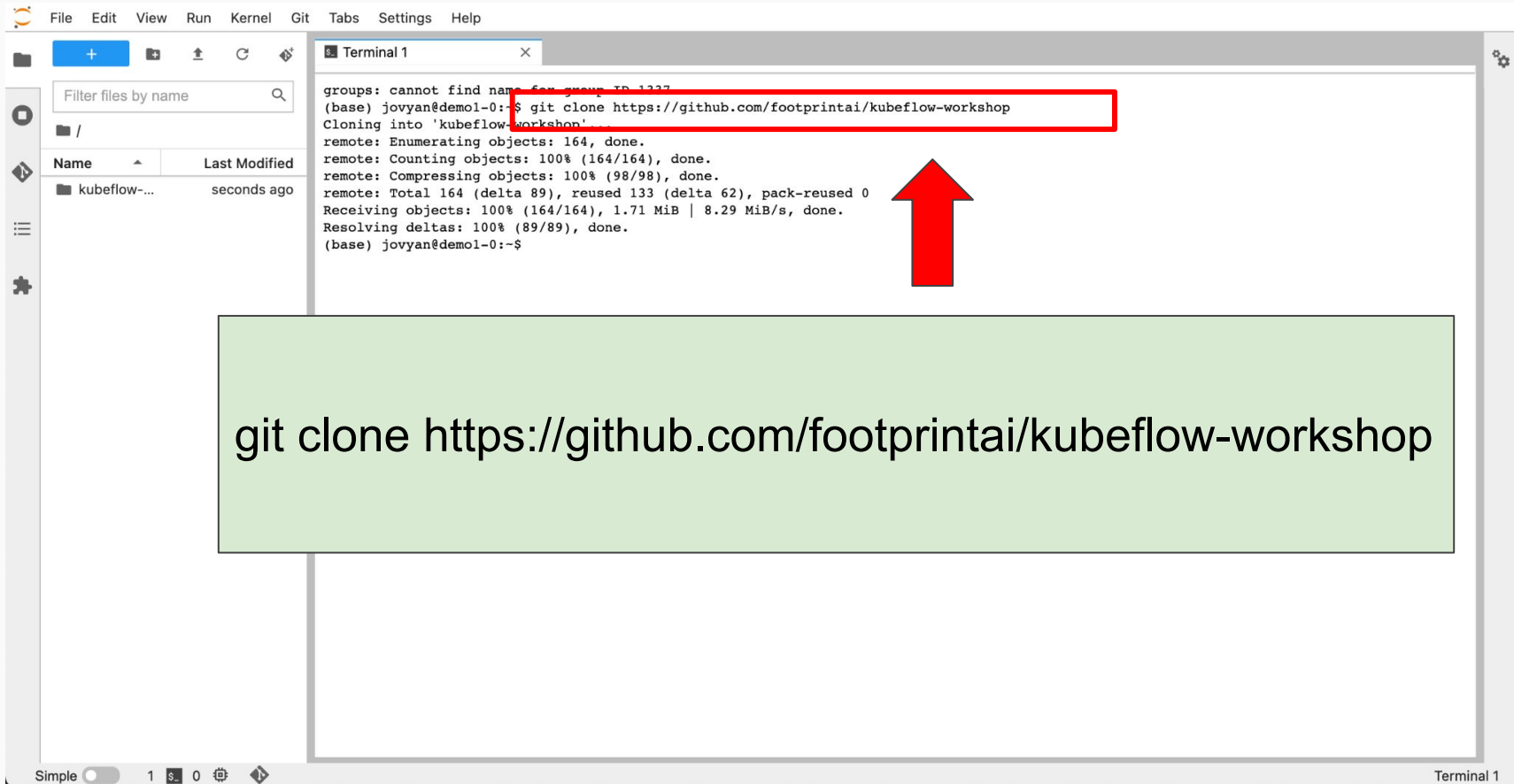
Status	Name	Type	Age	Image	GPUs	CPUs	Memory	Volumes	
✓	demo1		20 hours ago	jupyter-scipy:v1.4	0	0.5	1Gi	⋮	<a href="#">CONNECT</a>  
✓	demo2		2 hours ago	jupyter-tensorflow-full:v1.4	0	0.5	1Gi		<a href="#">CONNECT</a>  



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



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

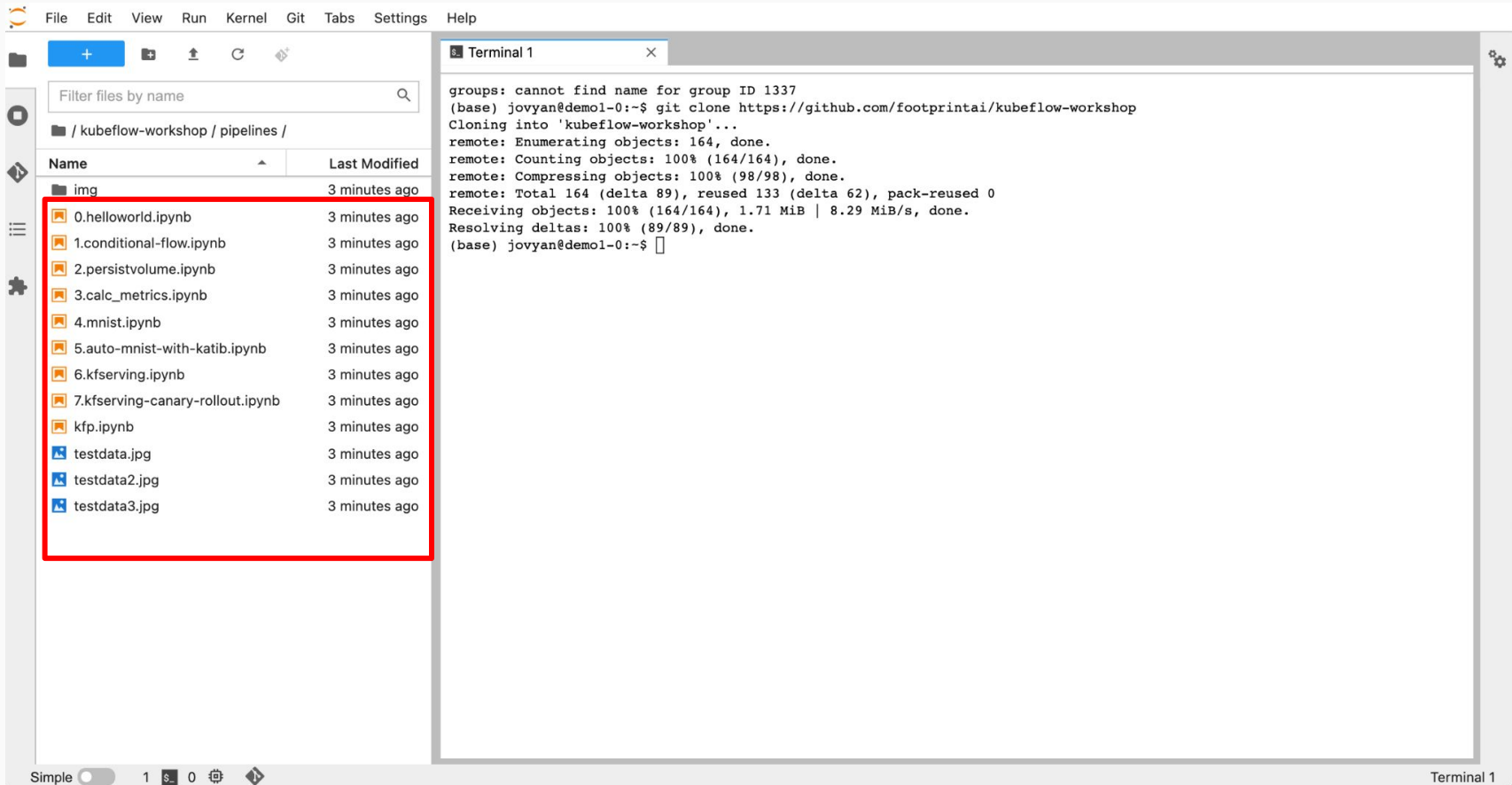


The screenshot shows a JupyterLab interface with a terminal window titled "Terminal 1". The terminal displays the output of a `git clone` command. A red rectangular box highlights the command `(base) jovyan@demo1-0:~$ git clone https://github.com/footprintai/kubeflow-workshop`. A large red arrow points from a green box below the terminal to this command. The green box contains the text `git clone https://github.com/footprintai/kubeflow-workshop`. The terminal output shows the progress of cloning the repository, including enumerating objects, counting objects, compressing objects, and receiving objects. The output ends with `(base) jovyan@demo1-0:~$`.

```
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: `/ kubeflow-workshop / pipelines /`. A red box highlights a group of files, including `img` and several Jupyter Notebook files (ipynb) and image files (jpg).

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

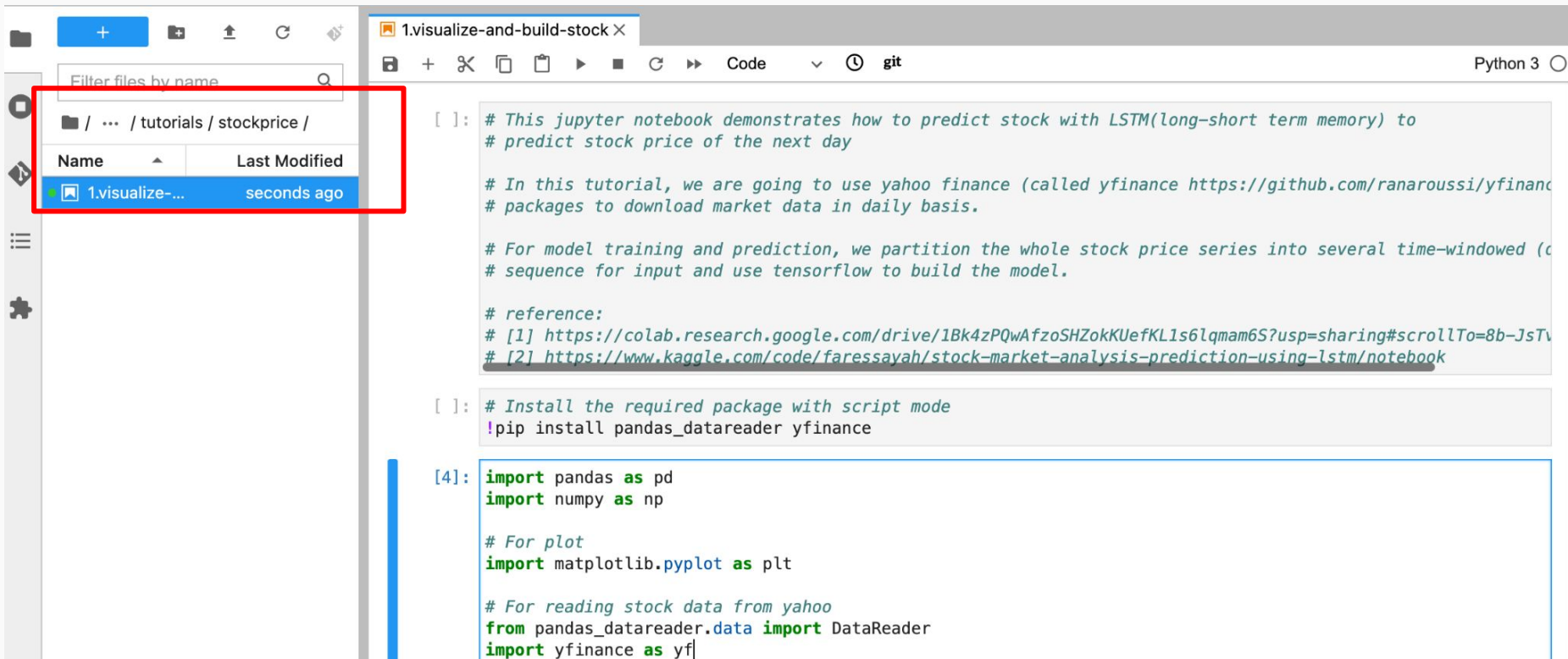
On the right, the terminal window shows the output of a `git clone` command:

```
(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



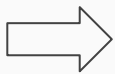
## Step3: Open financial model ipynb under tutorials/stockprice



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 its last modified time shown as `seconds ago`. The main area shows the code editor for this file, which contains 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

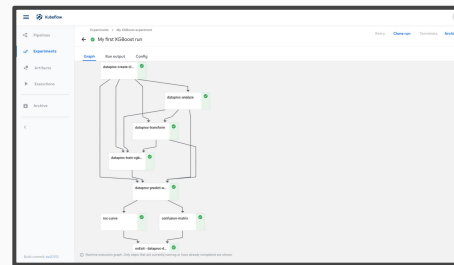
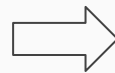
[illegible]

## Pipeline Code

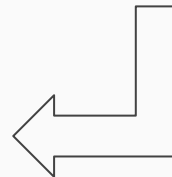
Compiled

## Workflow Resource

## Create a Pipeline



## Create Run



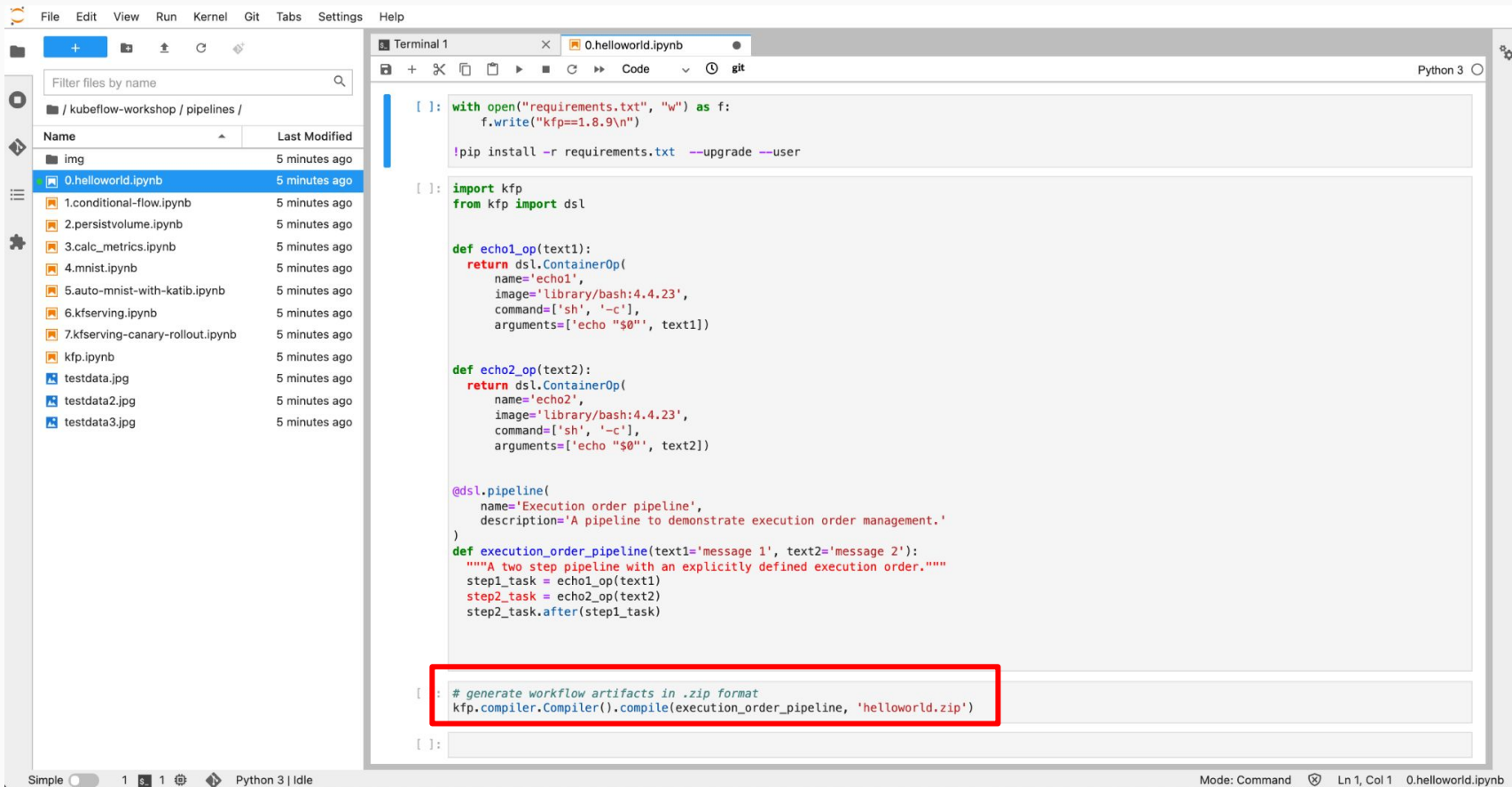
## Experiment1

Experiment2

## Runs

# Hello World Example

## Step4: Compile helloworld.ipynb (1/2)



The screenshot displays the JupyterLab environment. On the left, the file explorer shows a directory structure with files like '1.conditional-flow.ipynb', '2.persistvolume.ipynb', etc., and '0.helloworld.ipynb' is currently selected. The main area shows the code from '0.helloworld.ipynb'. The code defines a pipeline with two tasks, 'echo1\_op' and 'echo2\_op', and a function 'execution\_order\_pipeline'. A red box highlights the final line of code, which generates workflow artifacts in .zip format using the KFP compiler.

```
[ ]: with open("requirements.txt", "w") as f:
    f.write("kfp==1.8.9\n")

!pip install -r requirements.txt --upgrade --user

[ ]: import kfp
    from kfp import dsl

    def echo1_op(text1):
        return dsl.ContainerOp(
            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])

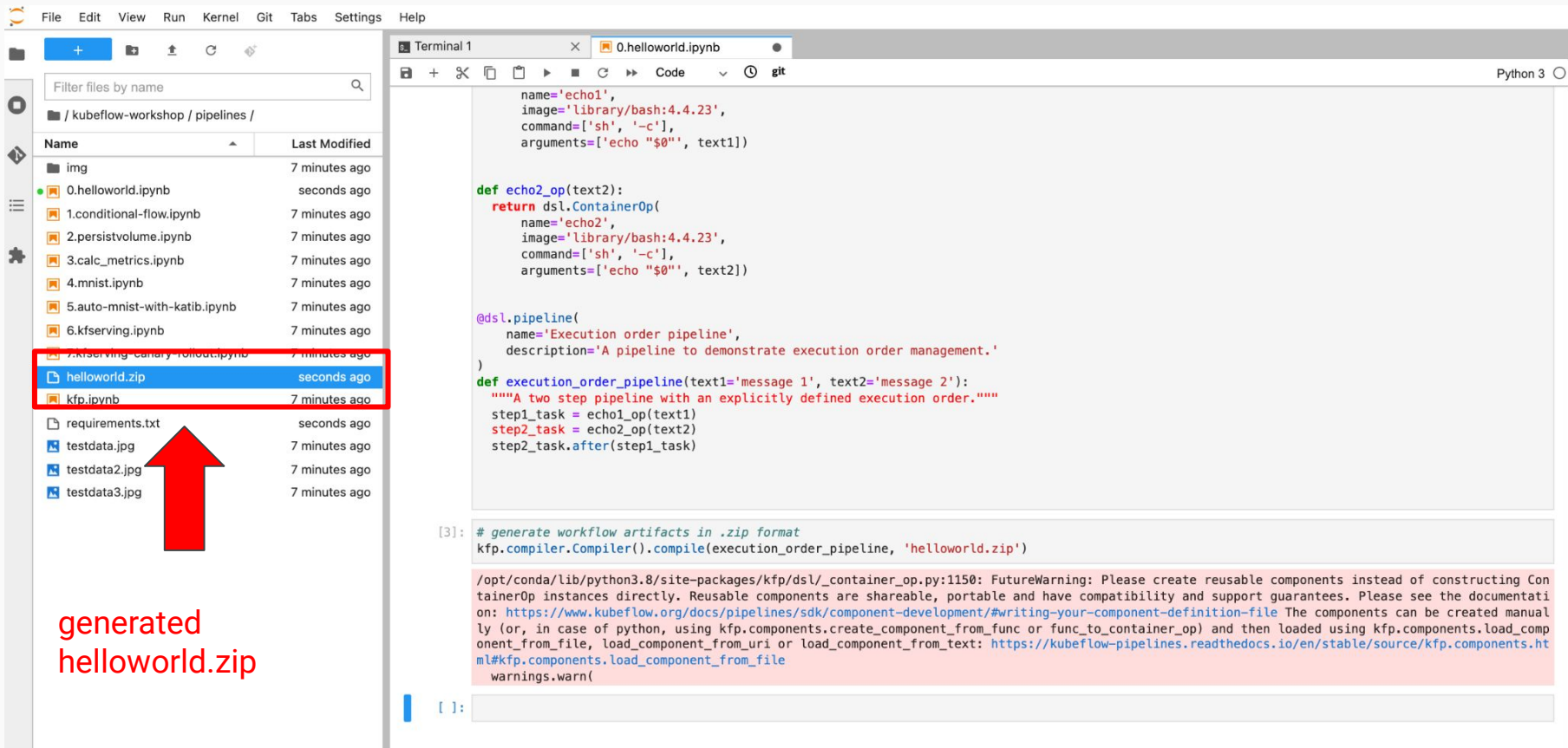
    @dsl.pipeline(
        name='Execution order pipeline',
        description='A pipeline to demonstrate execution order management.'
    )

    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)

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

[ ]:
```

## 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
<b>helloworld.zip</b>	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

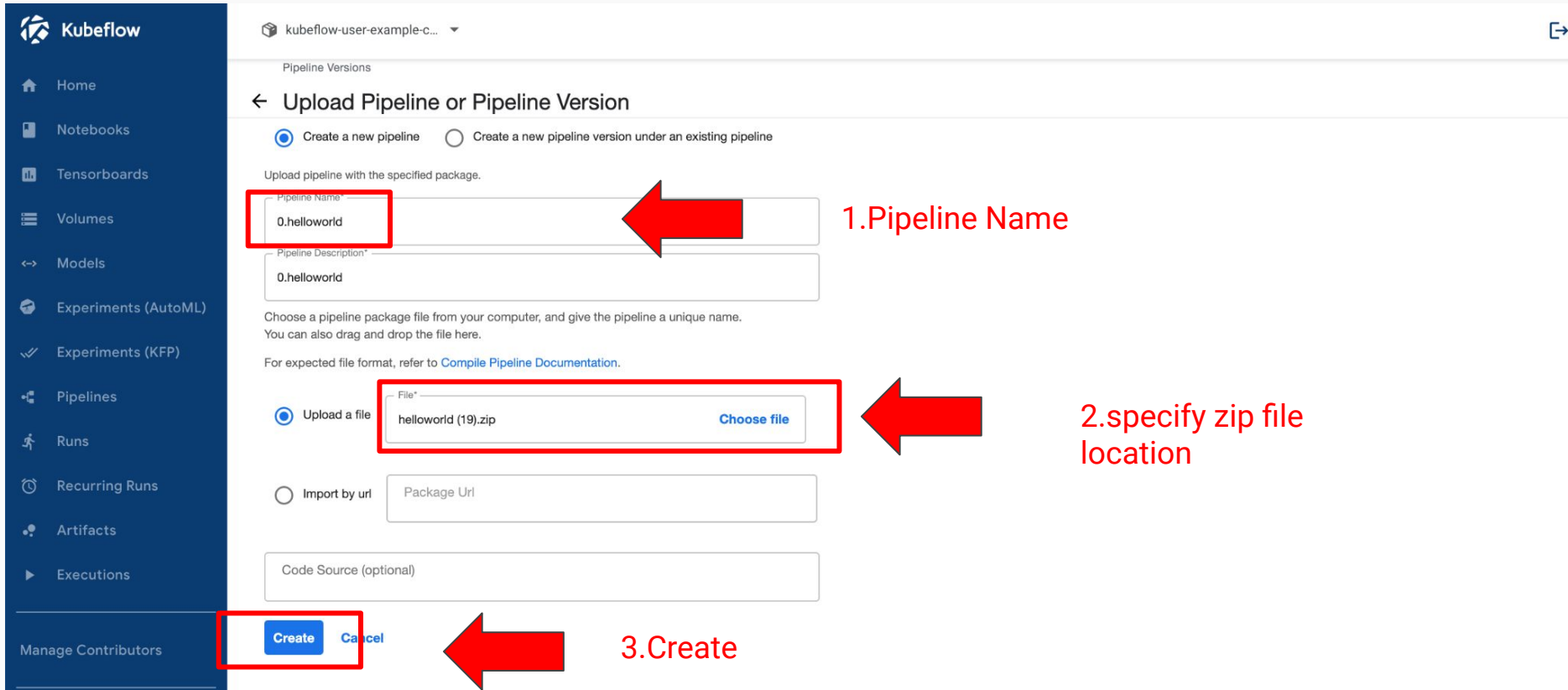
Delete

Filter pipelines

<input type="checkbox"/>	Pipeline name	Description	Upload
<input type="checkbox"/>	<a href="#">▶ [Tutorial] V2 lightweight Python com...</a>	<a href="#">source code</a> Shows different component input and output options for KFP v2 components.	11/30/2021, 1:02:25 PM
<input type="checkbox"/>	<a href="#">▶ [Tutorial] DSL - Control structures</a>	<a href="#">source code</a> 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"/>	<a href="#">▶ [Tutorial] Data passing in python co...</a>	<a href="#">source code</a> Shows how to pass data between python components.	11/30/2021, 1:02:23 PM
<input type="checkbox"/>	<a href="#">▶ [Demo] TFX - Taxi tip prediction mod...</a>	<a href="#">source code</a> <a href="#">GCP Permission requirements</a> . Example pipeline that does classification with model analysis based on...	11/30/2021, 1:02:22 PM
<input type="checkbox"/>	<a href="#">▶ [Demo] XGBoost - Iterative model tra...</a>	<a href="#">source code</a> 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'. The '+ Create experiment' button is highlighted with a red rectangular box. A large red arrow points upwards from the text 'Create an experiment' towards this button. Below the breadcrumb, there are tabs for 'Graph' and 'YAML'. The 'Graph' tab is active, showing a pipeline diagram with two steps: 'echo1' and 'echo2', connected by a downward arrow. A 'Simplify Graph' toggle is visible above the diagram. At the bottom of the main area is a 'Summary' panel with a 'Hide' link. The summary includes the ID '6f25028f-01e3-4acd-9389-7ec2031fb04b', the version '0.helloworld' with a dropdown arrow, and the 'Version source'.

Kubeflow

kubeflow-user-example-c...

Pipelines

← 0.helloworld (0.helloworld)

+ Create run + Upload version + Create experiment Delete

Graph YAML

Simplify Graph

echo1

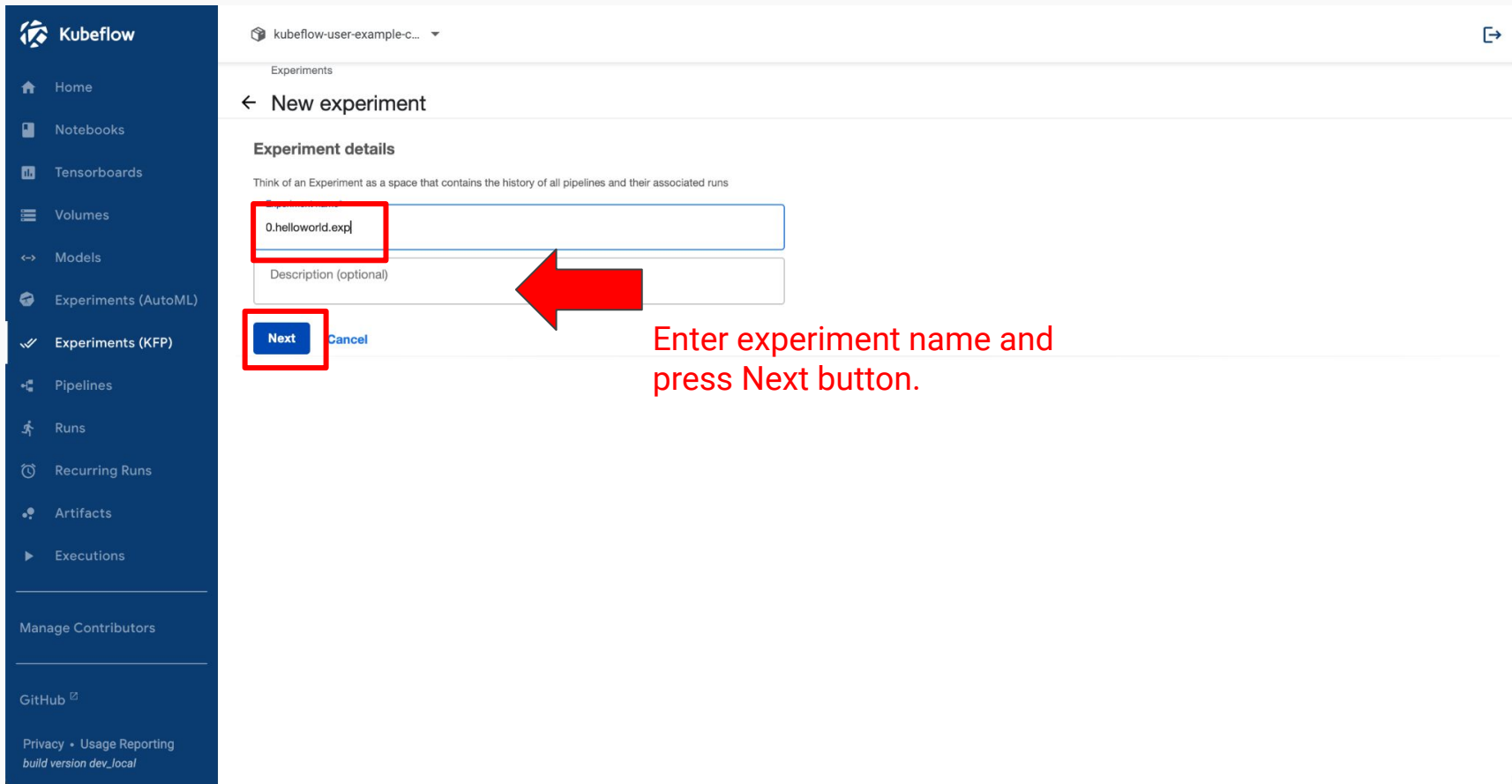
echo2

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-header '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 large red arrow points from the red 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 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

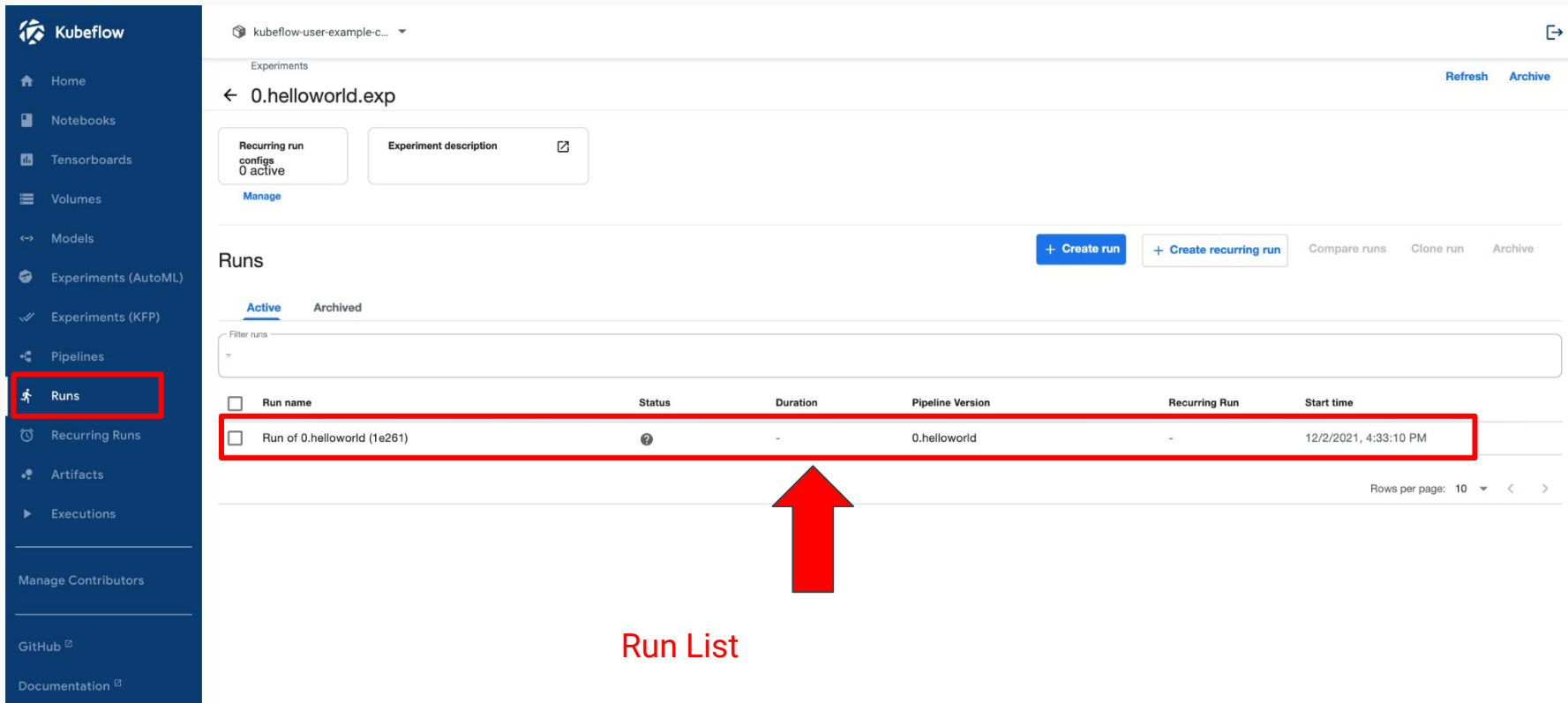
Start

Skip this step

1. Run a pipeline and specify its version

2. Add its experiment name

## Step4: Create a Pipeline (6/7)



The screenshot shows the Kubeflow web interface. On the left is a dark blue sidebar with navigation links: Home, Notebooks, Tensorboards, Volumes, Models, Experiments (AutoML), Experiments (KFP), Pipelines, **Runs** (highlighted with a red box), Recurring Runs, Artifacts, Executions, Manage Contributors, GitHub, and Documentation. The main content area is titled 'Experiments' and shows an experiment named '0.helloworld.exp'. Below this, there are buttons for 'Recurring run configs 0 active' and 'Experiment description'. The 'Runs' section is active, showing a table of runs. The table has columns: Run name, Status, Duration, Pipeline Version, Recurring Run, and Start time. A single row is visible, representing a run of '0.helloworld' with a status of '?' and a start time of '12/2/2021, 4:33:10 PM'. This row is highlighted with a red box. A large red arrow points from the text 'Run List' below the table to the highlighted row. At the bottom right, it says 'Rows per page: 10'.

Kubeflow

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

Experiments

kubeflow-user-example-c...

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)' under the 'Experiments > 0.helloworld.exp' path. It includes tabs for 'Graph', 'Run output', and 'Config'. The 'Graph' tab shows a pipeline graph with two steps, 'echo1' and 'echo2', connected by a downward arrow. Both steps have green checkmarks and refresh icons. A red rectangle highlights this graph. To the right, a modal window titled 'execution-order-pipeline-fxxfn-4223123588' is open, showing the 'Input/Output' tab. This tab 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 an entry for 'main-logs' with a link to 'minio://mlpipeline/artifacts/execution-order-pipeline-qvlt5/2021/11/30/execution-order-pipeline-qvlt5-137025384/main.log' and a 'View All' link. A large red arrow points from the text 'Running outputs' to the 'main-logs' artifact entry.

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

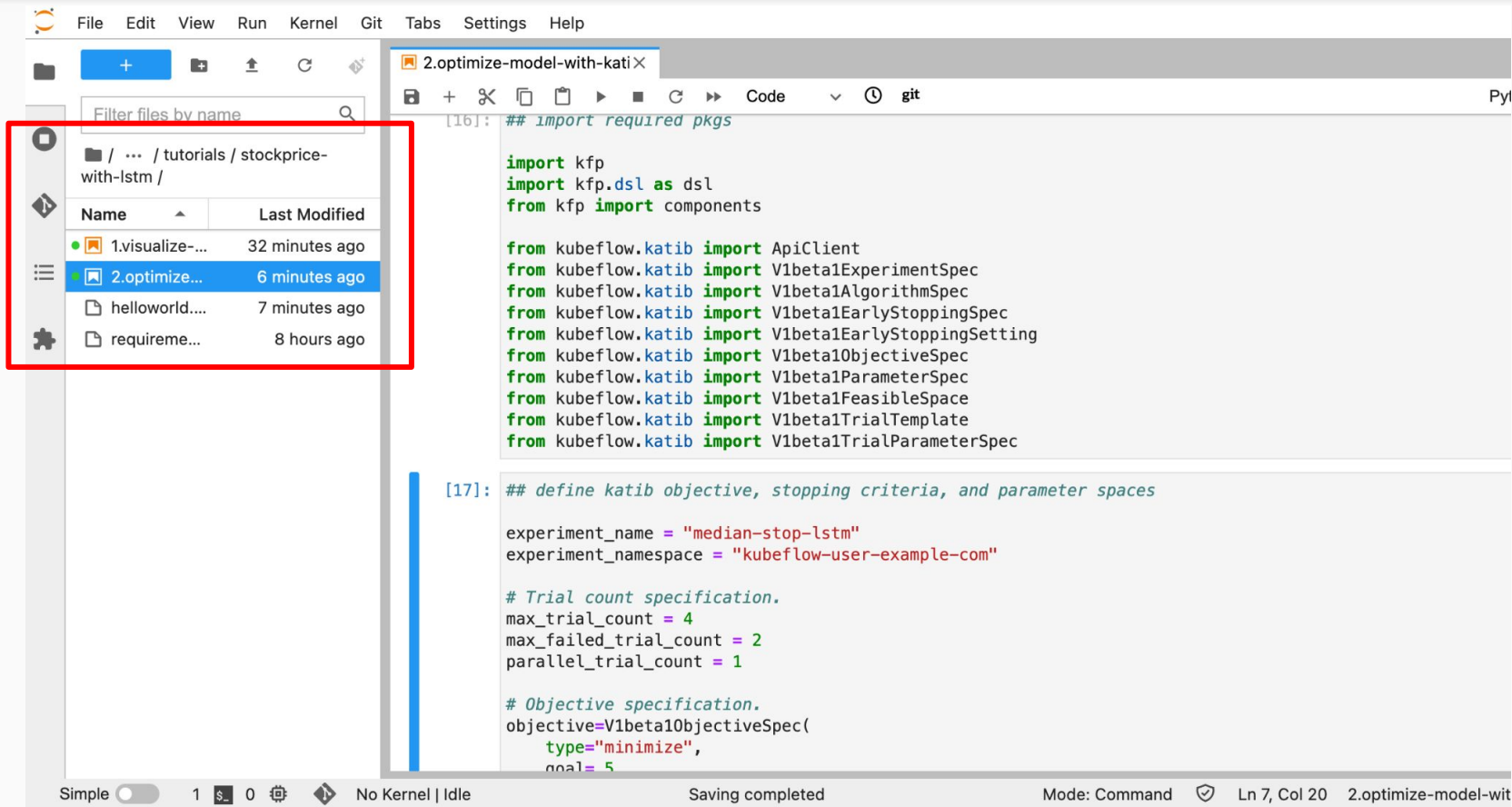
main-logs minio://mlpipeline/artifacts/execution-order-pipeline-qvlt5/2021/11/30/execution-order-pipeline-qvlt5-137025384/main.log View All

message 1

Running outputs

# Hyperparameter Example

## Step5: Hyperparameter tuning with katib (1/4)



The screenshot displays the JupyterLab interface. On the left, the file explorer shows the directory structure: `/ ... / tutorials / stockprice-with-lstm /`. A red box highlights the file list, which includes:

Name	Last Modified
1.visualize-...	32 minutes ago
2.optimize-...	6 minutes ago
helloworld....	7 minutes ago
requireme...	8 hours ago

The main code editor shows the following Python code:

```
[16]: ## import required pkgs

import kfp
import kfp.dsl as dsl
from kfp import components

from kubeflow.katib import ApiClient
from kubeflow.katib import V1beta1ExperimentSpec
from kubeflow.katib import V1beta1AlgorithmSpec
from kubeflow.katib import V1beta1EarlyStoppingSpec
from kubeflow.katib import V1beta1EarlyStoppingSetting
from kubeflow.katib import V1beta1ObjectiveSpec
from kubeflow.katib import V1beta1ParameterSpec
from kubeflow.katib import V1beta1FeasibleSpace
from kubeflow.katib import V1beta1TrialTemplate
from kubeflow.katib import V1beta1TrialParameterSpec

[17]: ## define katib objective, stopping criteria, and parameter spaces


experiment_name = "median-stop-lstm"
experiment_namespace = "kubeflow-user-example-com"

# Trial count specification.
max_trial_count = 4
max_failed_trial_count = 2
parallel_trial_count = 1


# Objective specification.
objective=V1beta1ObjectiveSpec(
    type="minimize",
    goal= 5
```

The status bar at the bottom indicates: Simple, 1, 0, No Kernel | Idle, Saving completed, Mode: Command, Ln 7, Col 20, 2.optimize-model-wit.

## Step5: Hyperparameter tuning with katib (2/4)

 **Kubeflow**


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

kubeflow-user-example-c... 

Experiments [Refresh](#) [Archive](#)

[<](#) hello1

Recurring run configs  
0 active  
[Manage](#)

Experiment description 

[+ Create run](#) [+ Create recurring run](#) [Compare runs](#) [Clone run](#) [Archive](#)

Runs

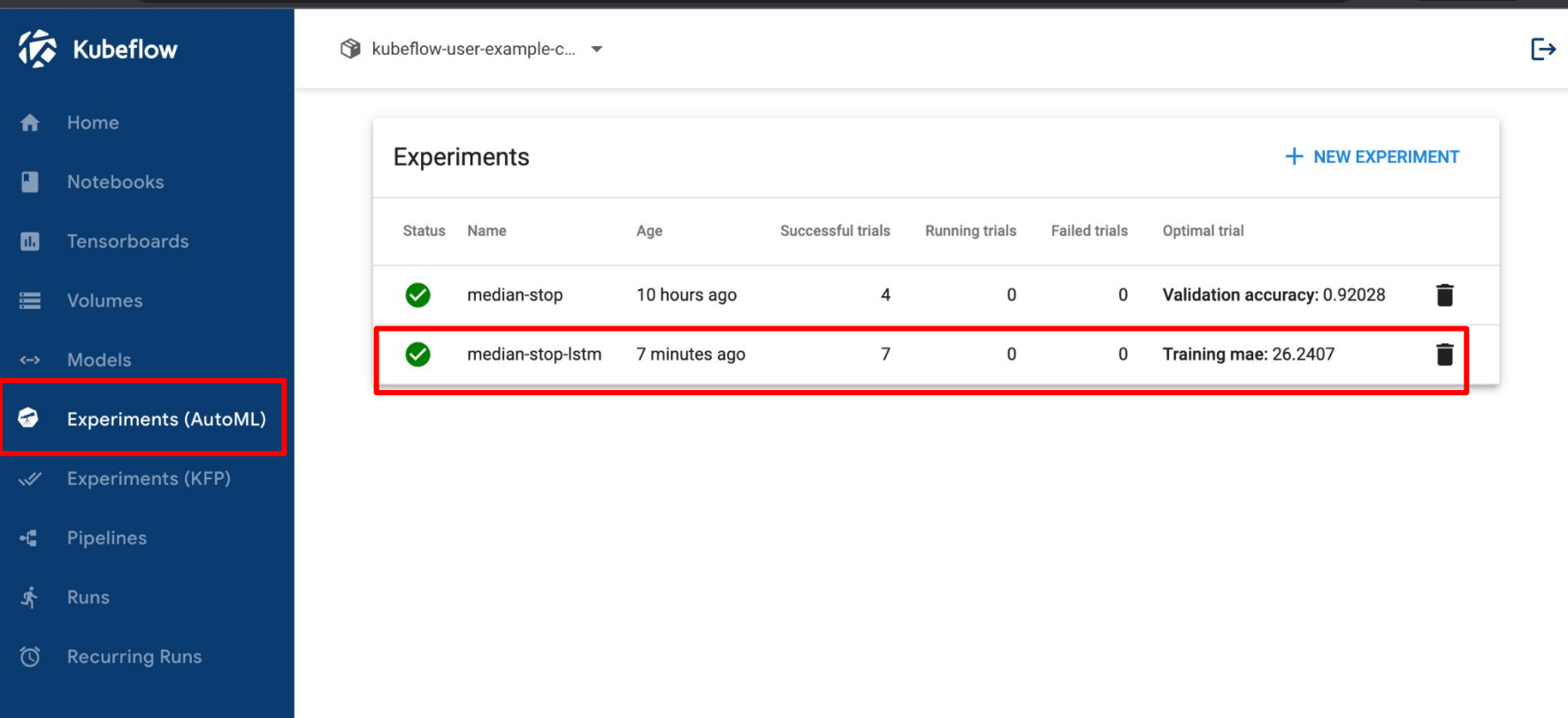
[Active](#) [Archived](#)

Filter runs

<input type="checkbox"/>	Run name	Status	Duration	Pipeline Version	Recurri...	Start time	quotient	remainder
<input type="checkbox"/>	Run of hello-world_versio...	?	-	hello-world_vers...	-	5/17/2022, 9:42:...		
<input type="checkbox"/>	Run of hello-world_versio...	✓	0:06:19	hello-world_vers...	-	5/17/2022, 9:30:...		
<input type="checkbox"/>	Run of hello-world_versio...	✓	0:01:24	hello-world_vers...	-	5/17/2022, 9:23:...	0.000	6.000



## Step5: Hyperparameter tuning with katib (3/4)



The image shows the Kubeflow Experiments (AutoML) interface. On the left is a dark blue sidebar with navigation links. The main area displays a table of experiments under the heading 'Experiments'. A red rectangle highlights the second row of the table, which represents the 'median-stop-lstm' experiment. This experiment is marked as successful with a green checkmark, started 7 minutes ago, and has 7 successful trials, 0 running trials, and 0 failed trials. The optimal trial metric is 'Training mae: 26.2407'.

**Kubeflow**

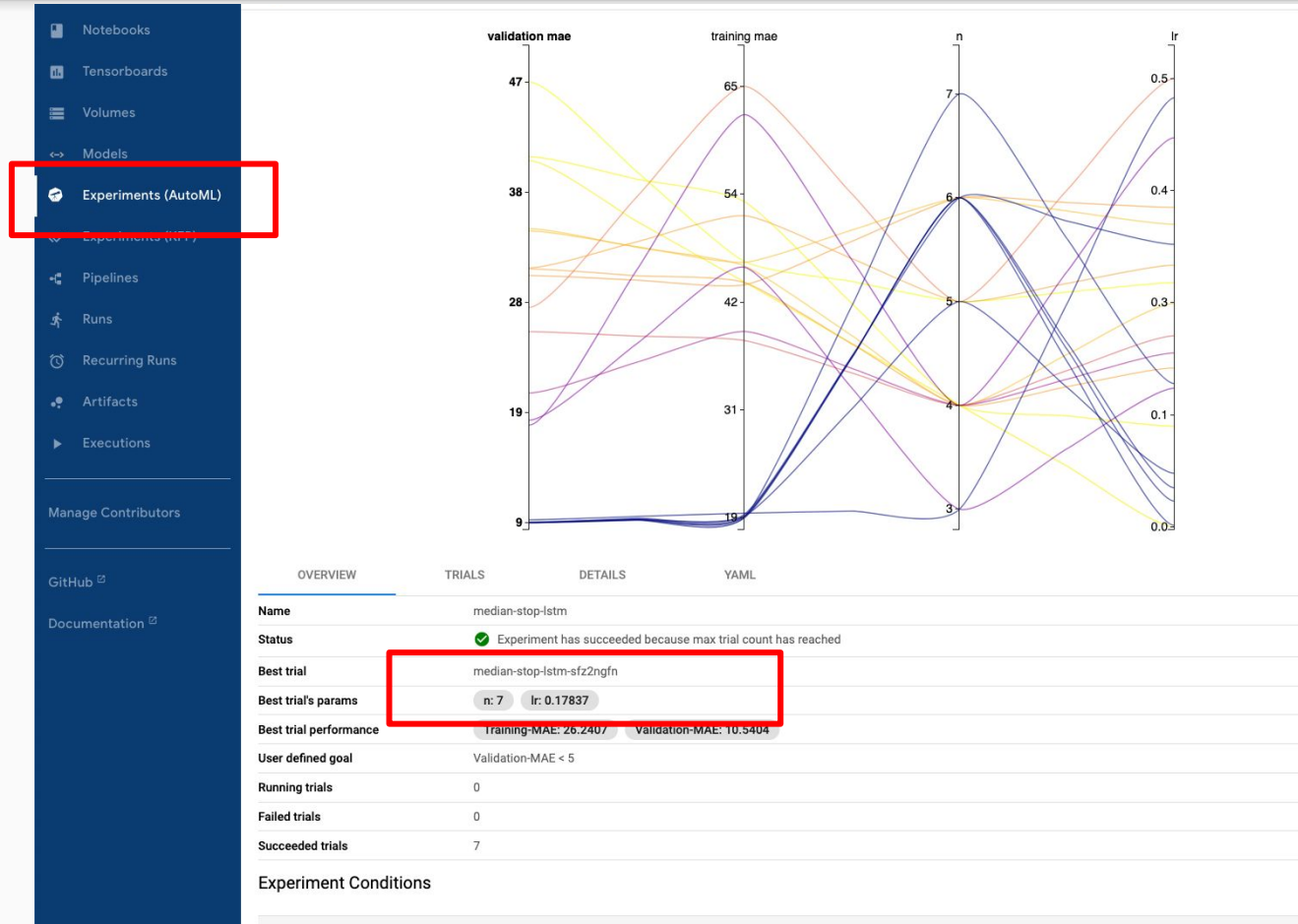
- Home
- Notebooks
- Tensorboards
- Volumes
- Models
- Experiments (AutoML)**
- Experiments (KFP)
- Pipelines
- Runs
- Recurring Runs

kubeflow-user-example-c... ▾

**Experiments** [+ NEW EXPERIMENT](#)

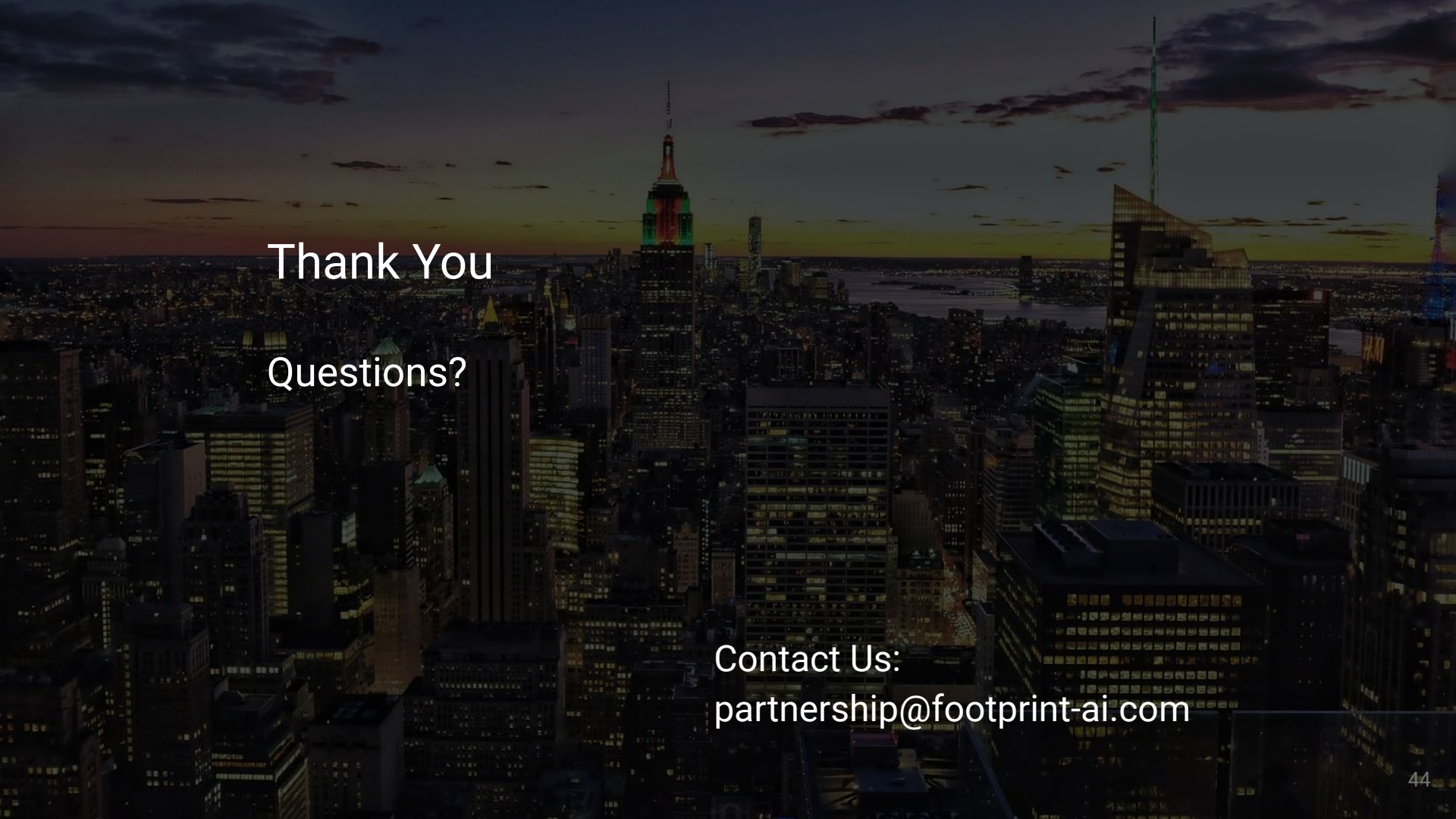
Status	Name	Age	Successful trials	Running trials	Failed trials	Optimal trial	
✓	median-stop	10 hours ago	4	0	0	Validation accuracy: 0.92028	🗑️
✓	median-stop-lstm	7 minutes ago	7	0	0	Training mae: 26.2407	🗑️

## Step5: Hyperparameter tuning with katib (4/4)



# One minute takeaway

- Implemented remote development, training, and deployment.
- Pipeline-based development model
- Distributed model training and optimization

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 State Thruway Bridge spanning it. The overall scene is a vibrant and detailed representation of the city at night.

Thank You  
Questions?

Contact Us:  
[partnership@footprint-ai.com](mailto:partnership@footprint-ai.com)