

Kubeflow Workshop

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

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



課程綱要

- 課前知識
- 概念簡介
- 環境介紹
- 詞彙定義
- 範例練習
- 問與答

課前知識

- Be comfortable with UNIX command line
 - Navigating directories with ``cd`` or ``tree``
 - Editing files, like ``vim``, ``nano``
 - Bash scripting, like env or looping
- Be an expert with ``Google``
 - <https://letmegoogletthat.com/?q=you+can+google+it>
- It is totally OK if you don't know what is Container and Kubernetes

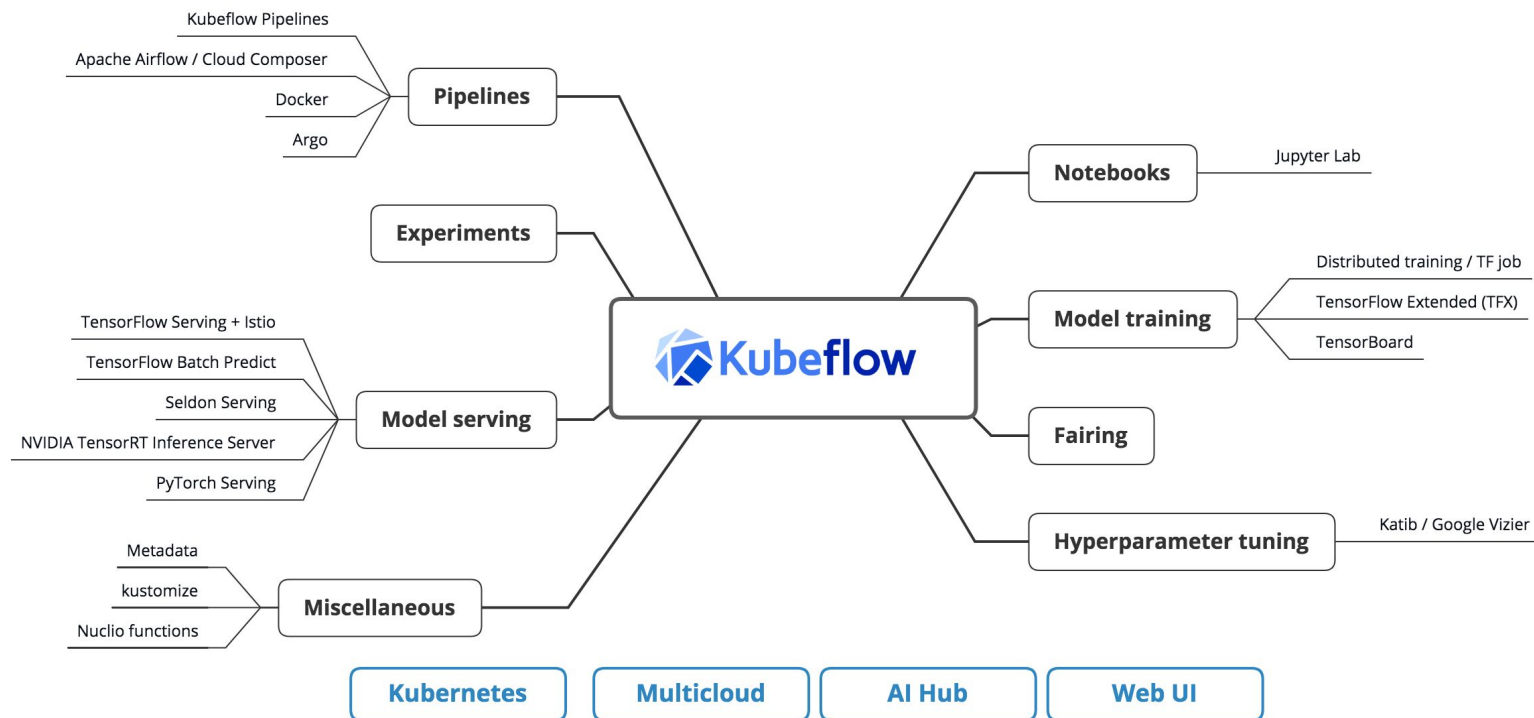
荀子《儒效篇》

「不聞不若聞之，聞之不若見之，見之不若知之，知之不若行之；學至于行之而止矣。」

```
git clone https://github.com/FootprintAI/kubeflow-workshop
```

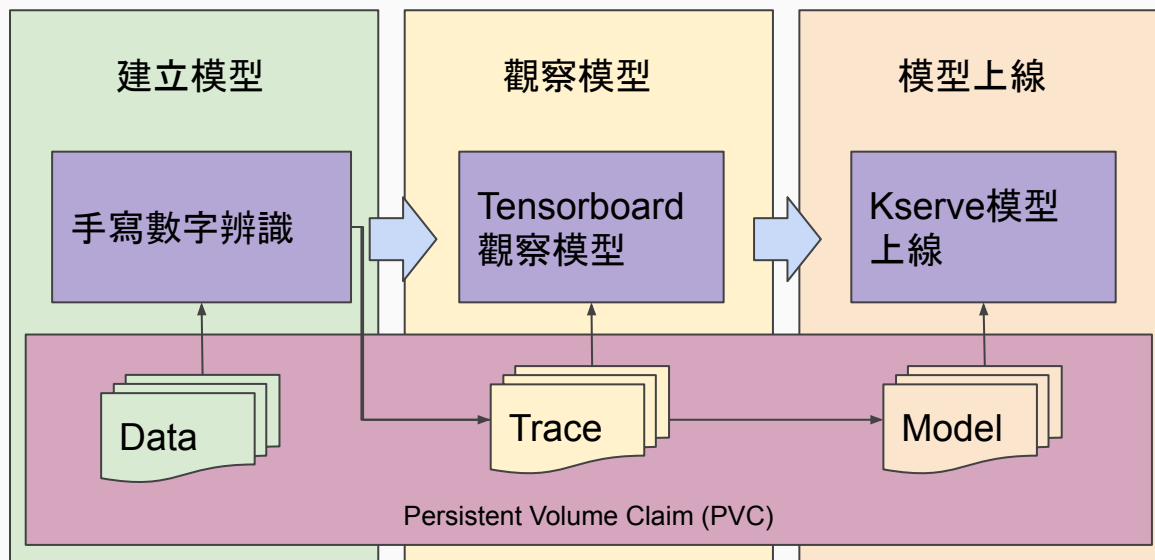
Or [Click Me](#)

Kubeflow架構



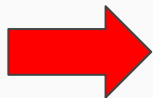
概念簡介

e2e machine learning flow

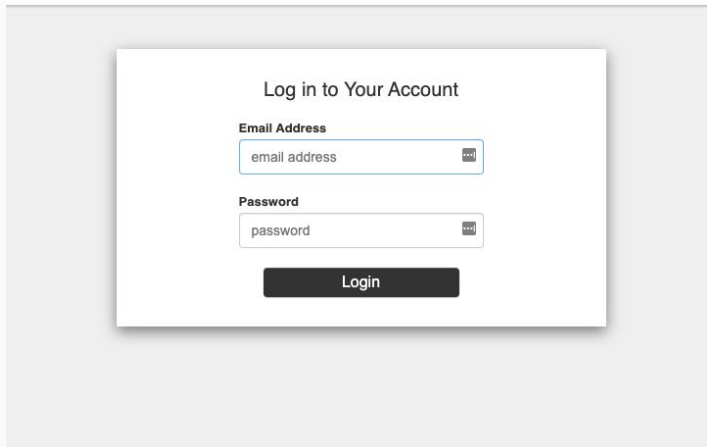
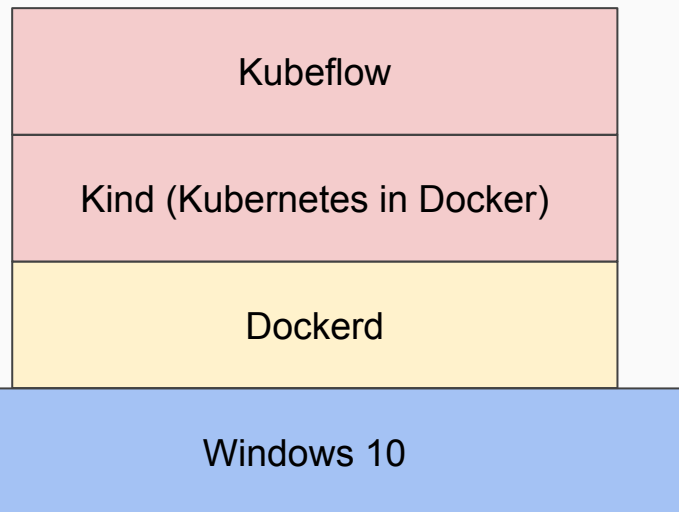
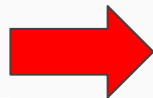


環境介紹

1. `kubectl port-forward
svc/istio-ingressgateway -n istio-system
8080:80 --address 0.0.0.0`



2. Open `http://localhost:8080`



Wait! 所以我說那個帳號密碼呢?

Account: user@example.com

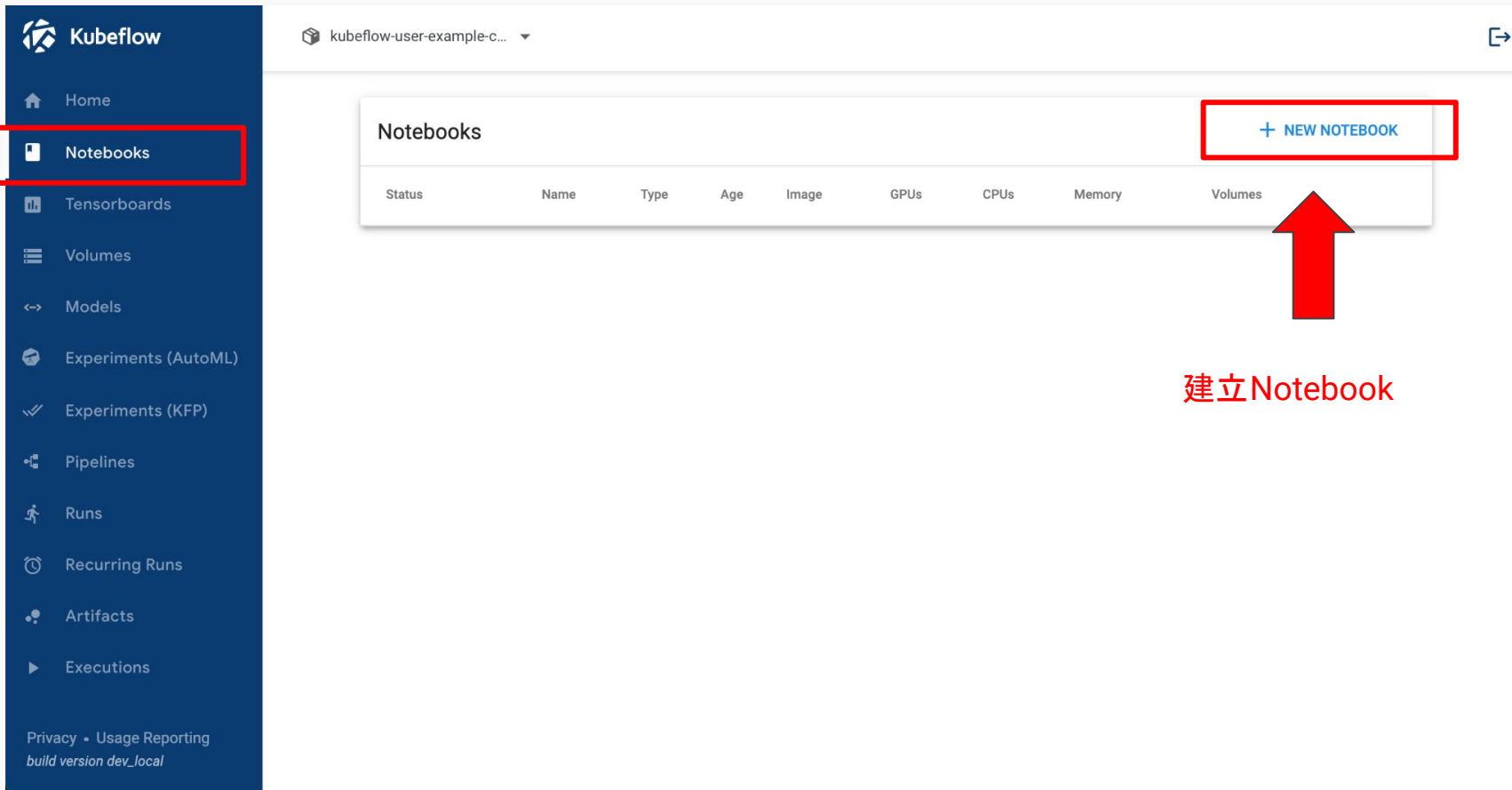
Password: 12341234

```
// 查看所有namespace  
Kubectl get namespaces
```

```
// 查看kubeflow中所有運行的Pod  
kubectl get pods -n kubeflow
```

```
// 查看目前使用者運行的Pod  
kubectl get pods -n kubeflow-user-example-com
```

Step1: 開啟Notebook作為開發環境 (1/3)



The screenshot displays the Kubeflow dashboard interface. On the left, a dark blue sidebar contains a list of navigation items: Home, Notebooks, Tensorboards, Volumes, Models, Experiments (AutoML), Experiments (KFP), Pipelines, Runs, Recurring Runs, Artifacts, and Executions. The 'Notebooks' item is highlighted with a red rectangular box. The main content area on the right shows a 'Notebooks' section with a table header including Status, Name, Type, Age, Image, GPUs, CPUs, Memory, and Volumes. A red rectangular box highlights the '+ NEW NOTEBOOK' button in the top right corner of this section. A large red arrow points from the text '建立Notebook' (Create Notebook) below the button to the button itself.

Kubeflow

kubeflow-user-example-c...

Notebooks

+ NEW NOTEBOOK

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

建立Notebook

Privacy • Usage Reporting
build version dev_local

Step1: 開啟Notebook作為開發環境 (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 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

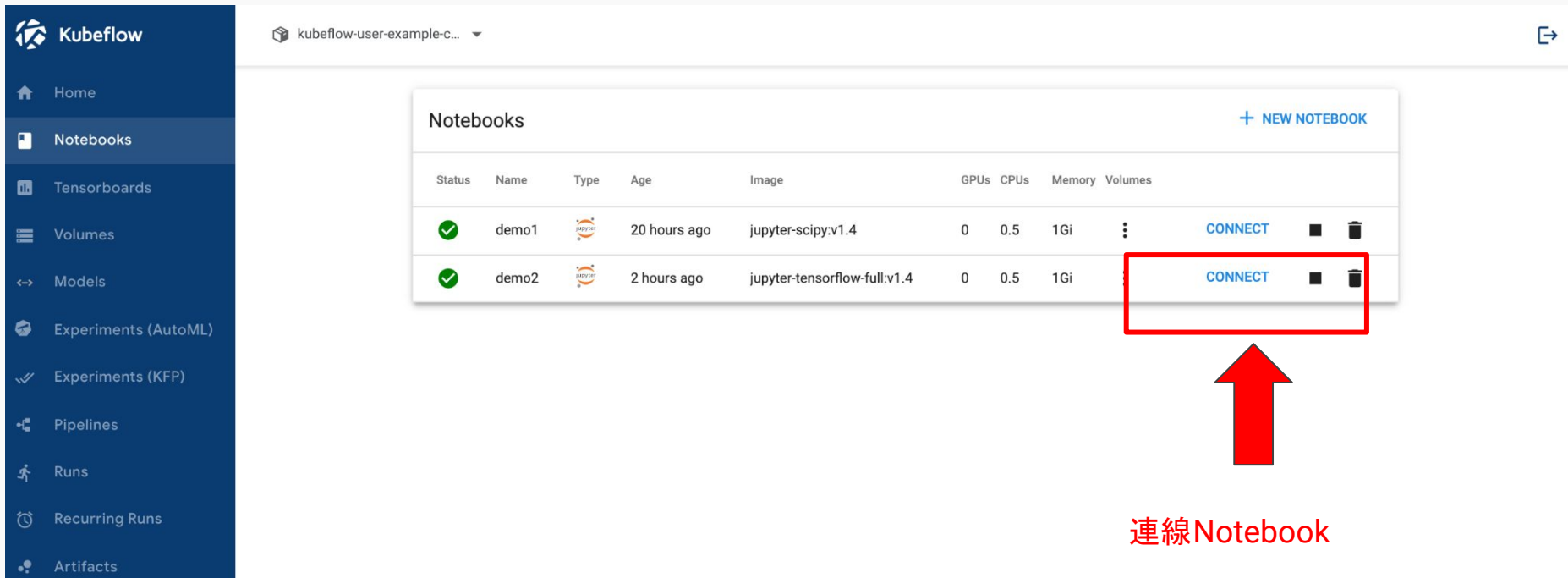
0.5

Requested memory in Gi







1

給定名稱以及指定其 CPU/Memory

Step1: 開啟Notebook作為開發環境 (3/3)

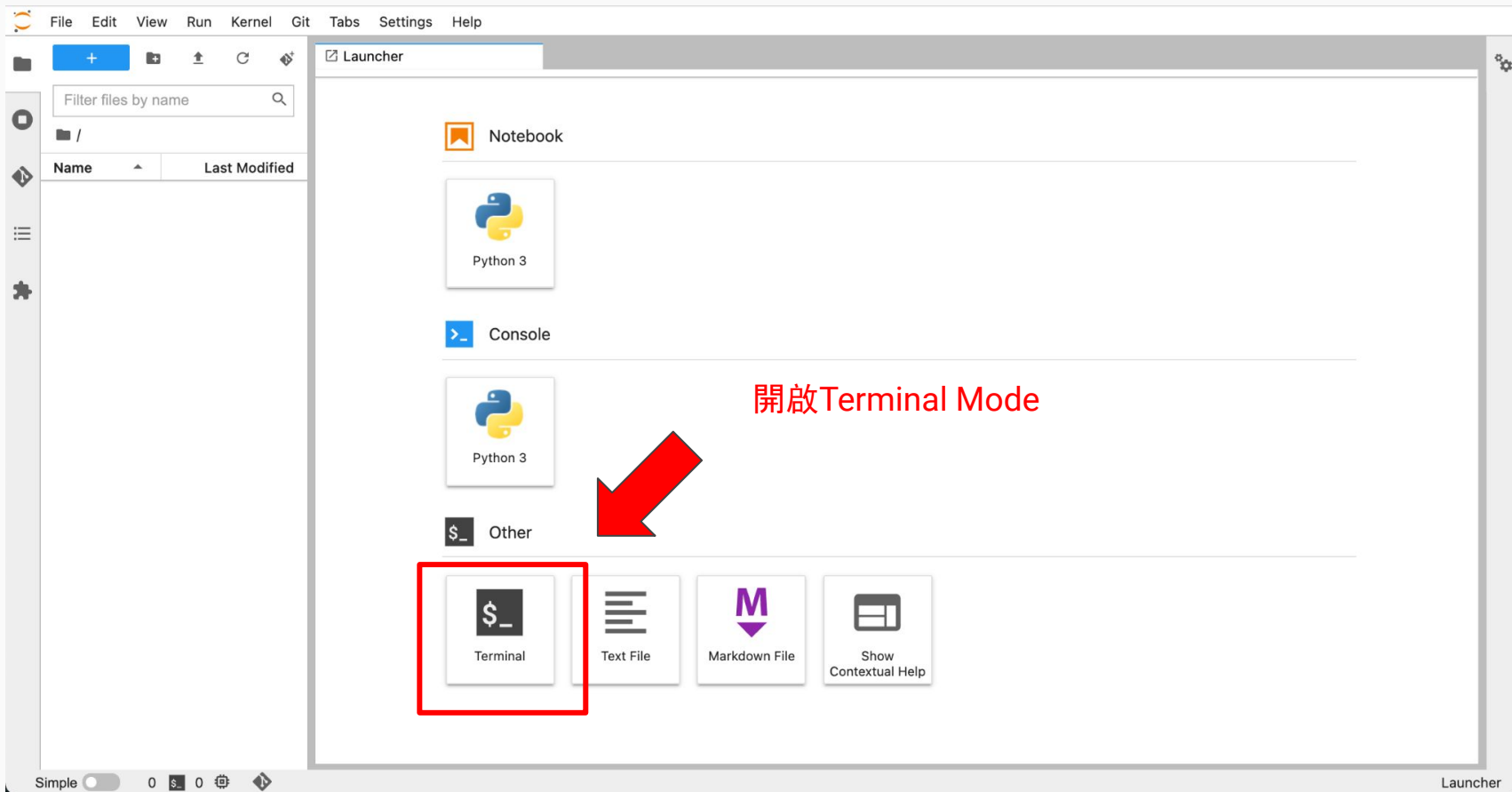


The image shows the Kubeflow web interface. On the left is a dark blue sidebar with navigation links: Home, Notebooks (selected), Tensorboards, Volumes, Models, Experiments (AutoML), Experiments (KFP), Pipelines, Runs, Recurring Runs, and Artifacts. The main content area has a header with a user dropdown 'kubeflow-user-example-c...' and a '+ NEW NOTEBOOK' button. Below is a 'Notebooks' table with columns: Status, Name, Type, Age, Image, GPUs, CPUs, Memory, Volumes, and actions. Two notebooks are listed: 'demo1' (jupyter-scipy:v1.4, 20 hours ago) and 'demo2' (jupyter-tensorflow-full:v1.4, 2 hours ago). The 'demo2' row's 'CONNECT' button is highlighted with a red box, and a large red arrow points from the text '連線Notebook' below to this button.

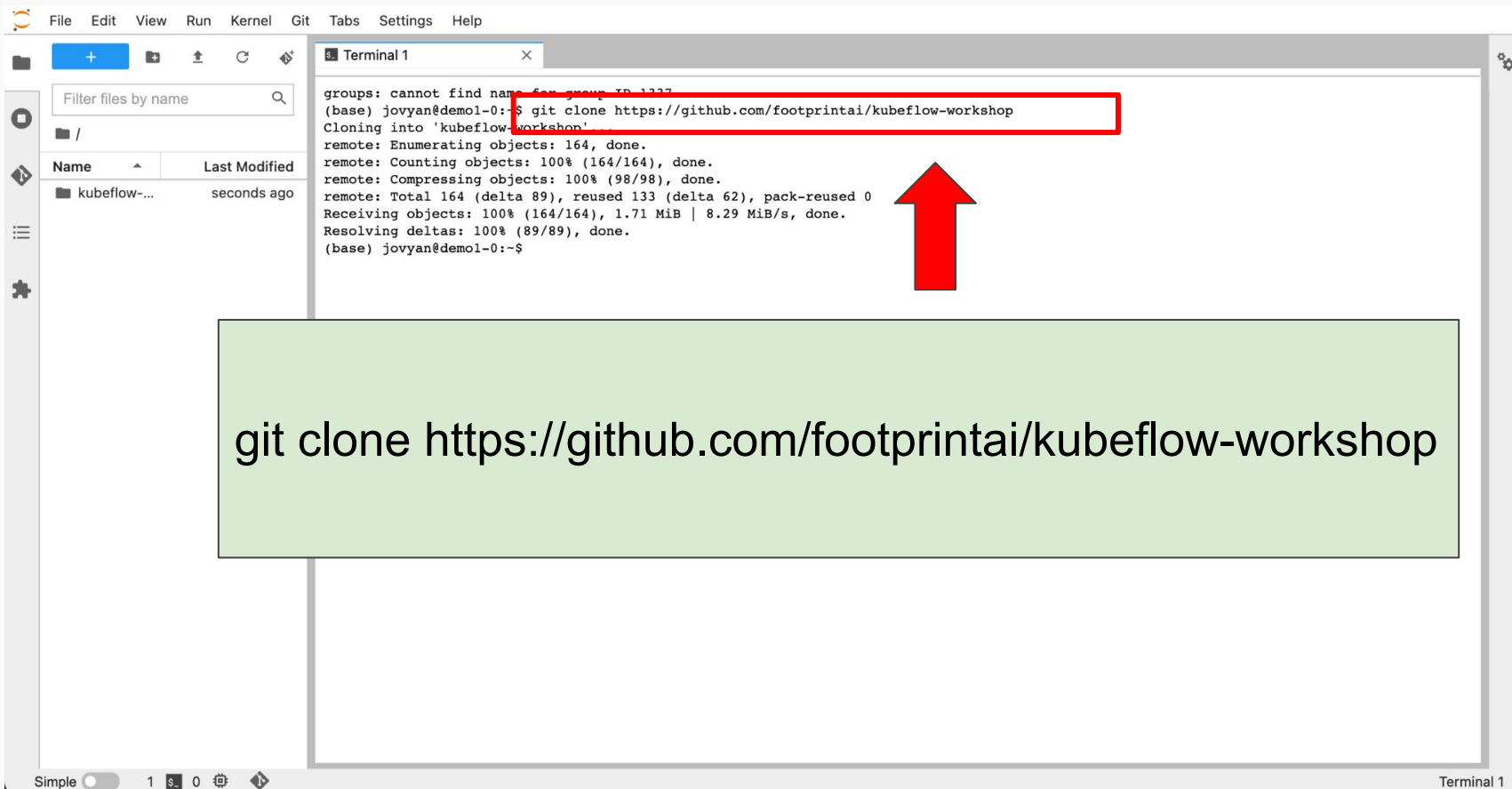
Status	Name	Type	Age	Image	GPUs	CPUs	Memory	Volumes	
✓	demo1		20 hours ago	jupyter-scipy:v1.4	0	0.5	1Gi	⋮	CONNECT  
✓	demo2		2 hours ago	jupyter-tensorflow-full:v1.4	0	0.5	1Gi		CONNECT  

連線Notebook

Step2: 開啟Terminal下載範例程式(1/3)



Step2: 開啟Terminal下載範例程式(2/3)



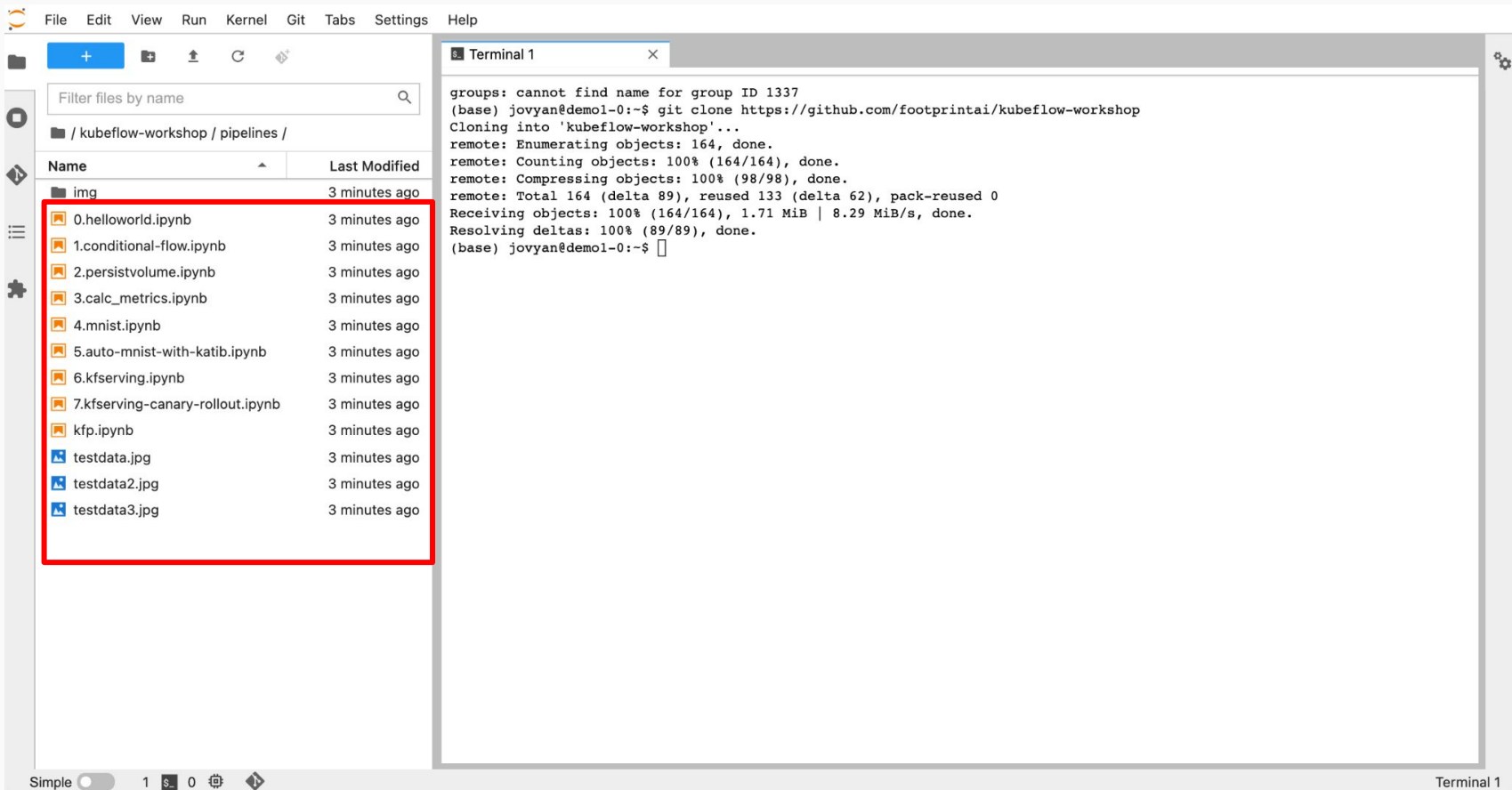
The screenshot shows a terminal window with the following output:

```
groups: cannot find name for group ID 1337
(base) joyvan@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) joyvan@demo1-0:~$
```

A red box highlights the command `git clone https://github.com/footprintai/kubeflow-workshop` in the terminal. A red arrow points from a large green box containing the same command text to the highlighted command in the terminal.

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

Step2: 開啟Terminal下載範例程式(3/3)



The screenshot shows a JupyterLab environment. On the left, the file browser displays the directory `/ kubeflow-workshop / pipelines /`. A red box highlights the following files:

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 (Terminal 1) 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:~$
```

詞彙說明



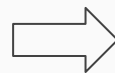
```
1 | with name="myworkflow-job", "C" as f {
2 |   workflowName: f.name
3 | }
4 |
5 | (pip install -r requirements.txt --upgrade --user)
6 |
7 | export K8S
8 |
9 | def main_workflow():
10 |   return all_containers()
11 |   name: "main"
12 |   image: "ubuntu:16.04.2"
13 |   command: ["ls", "-l"]
14 |   arguments: ["task", "main", "task"]
15 |
16 | def main_workflow():
17 |   return all_containers()
18 |   name: "main"
19 |   image: "ubuntu:16.04.2"
20 |   command: ["ls", "-l"]
21 |   arguments: ["task", "main", "task"]
22 |
23 | @cli.generate
24 | name: "Workflow order pipeline"
25 | description: "A pipeline to demonstrate execution order management."
26 | def execution_order_pipeline(message 1, task1 message 2) ...
27 |   """ We use pipeline with an explicitly defined execution order """
28 |   task1.task = main_workflow()
29 |   task2.task = main_workflow()
30 |   steps.task = ["task1", "task2"]
31 |
32 | # generate workflow artifacts in .job folder
33 | wf_compiler.Compiler().compile_workflow_resource_pipeline, "WorkflowJob.job"
```

Pipeline Code

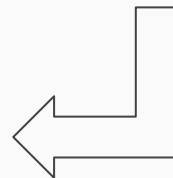
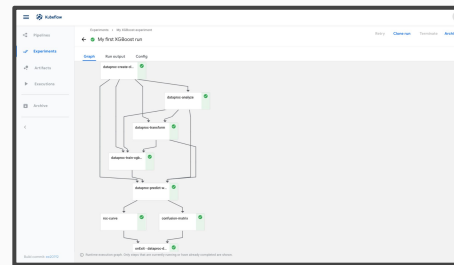


編譯後

Workflow
Resource



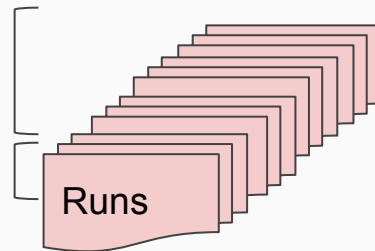
建立Pipeline



Create Run

實驗1 (Experiment1)

實驗2 (Experiment2)



範例1 Hello World!

Step3: 編譯helloworld.ipynb (1/2)

The screenshot shows the JupyterLab interface. On the left, a sidebar lists files in the 'kubeflow-workshop / pipelines /' directory. The file '0.helloworld.ipynb' is selected. The main editor displays the code from this file. The code defines a pipeline with two tasks, 'echo1_op' and 'echo2_op', and a function 'execution_order_pipeline' that uses these tasks. A red box highlights the final line of code, which generates a workflow artifact in .zip format.

```
[ ]: 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])

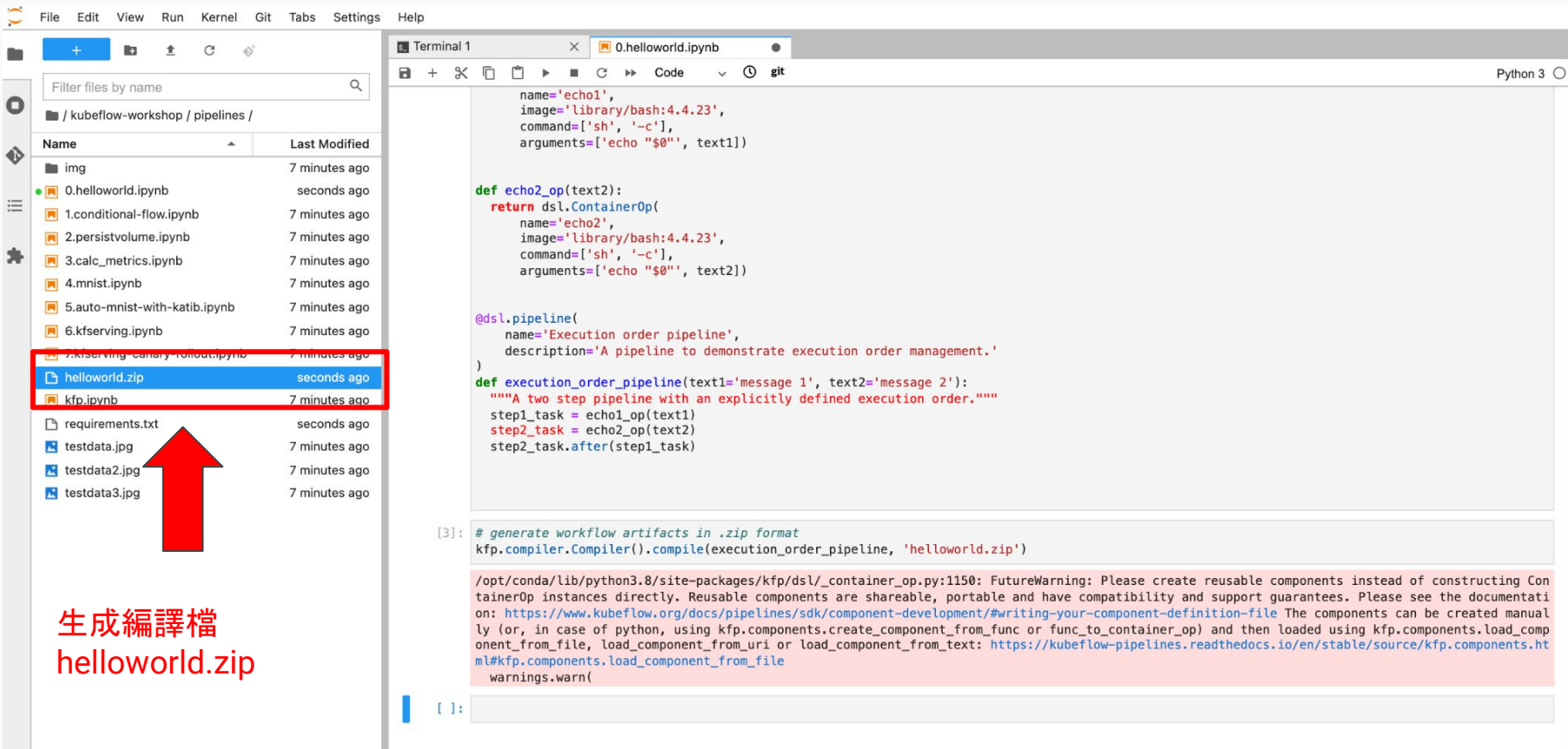
    @dsl.pipeline(
        name='Execution order pipeline',
        description='A pipeline to demonstrate execution order management.'
    )

    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')

[ ]:
```

Step3: 編譯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

生成編譯檔
helloworld.zip

```
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 KfpContainerOp 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: 建立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

Pipelines

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: 建立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**

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](#)

☐ Import by url **Package Url**

Code Source (optional)

Create [Cancel](#)

1.輸入Pipeline名稱

2.指定編譯完後 Zip位置

3.建立

Step4: 建立Pipeline (3/7)

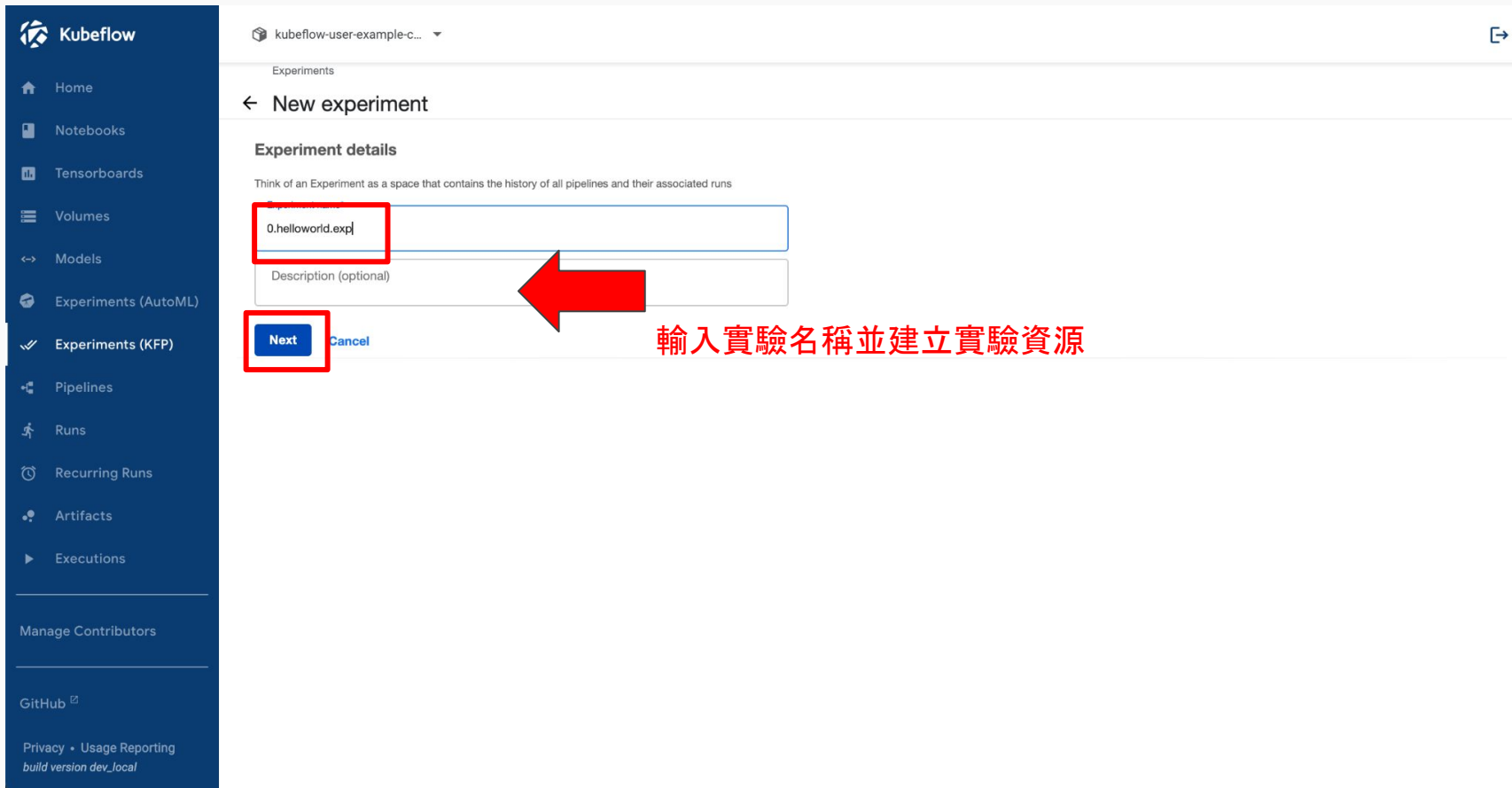
The screenshot displays the Kubeflow Pipelines web interface. On the left is a dark blue sidebar with navigation links: Home, Notebooks, Tensorboards, Volumes, Models, Experiments (AutoML), Experiments (KFP), Pipelines, Runs, Recurring Runs, Artifacts, and Executions. At the bottom of the sidebar is a link for 'Manage Contributors'.

The main content area shows the 'Pipelines' section for a user named 'kubeflow-user-example-c...'. The selected pipeline is '0.helloworld (0.helloworld)'. Above the pipeline graph 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 from the Chinese text '建立實驗以便管理 Run' (Create experiment to manage Run) below it to the highlighted button.

The pipeline graph is displayed in 'Graph' view, showing a 'Simplify Graph' toggle and a flow from 'echo1' to 'echo2'.

At the bottom, a 'Summary' panel is visible, showing the pipeline's ID (6f25028f-01e3-4acd-9389-7ec2031fb04b), version (0.helloworld), and version source.

Step4: 建立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) (which is 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 dropdown menu for 'kubeflow-user-example-c...'. Under the 'Experiment details' section, there is a text input field containing '0.helloworld.exp' and a 'Description (optional)' field below it. A red rectangular box highlights the text input field, and a red arrow points from the Chinese text '輸入實驗名稱並建立實驗資源' to the 'Next' button. The 'Next' button is also highlighted with a red rectangular box.

輸入實驗名稱並建立實驗資源

Step4: 建立Pipeline (5/7)

Kubeflow

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

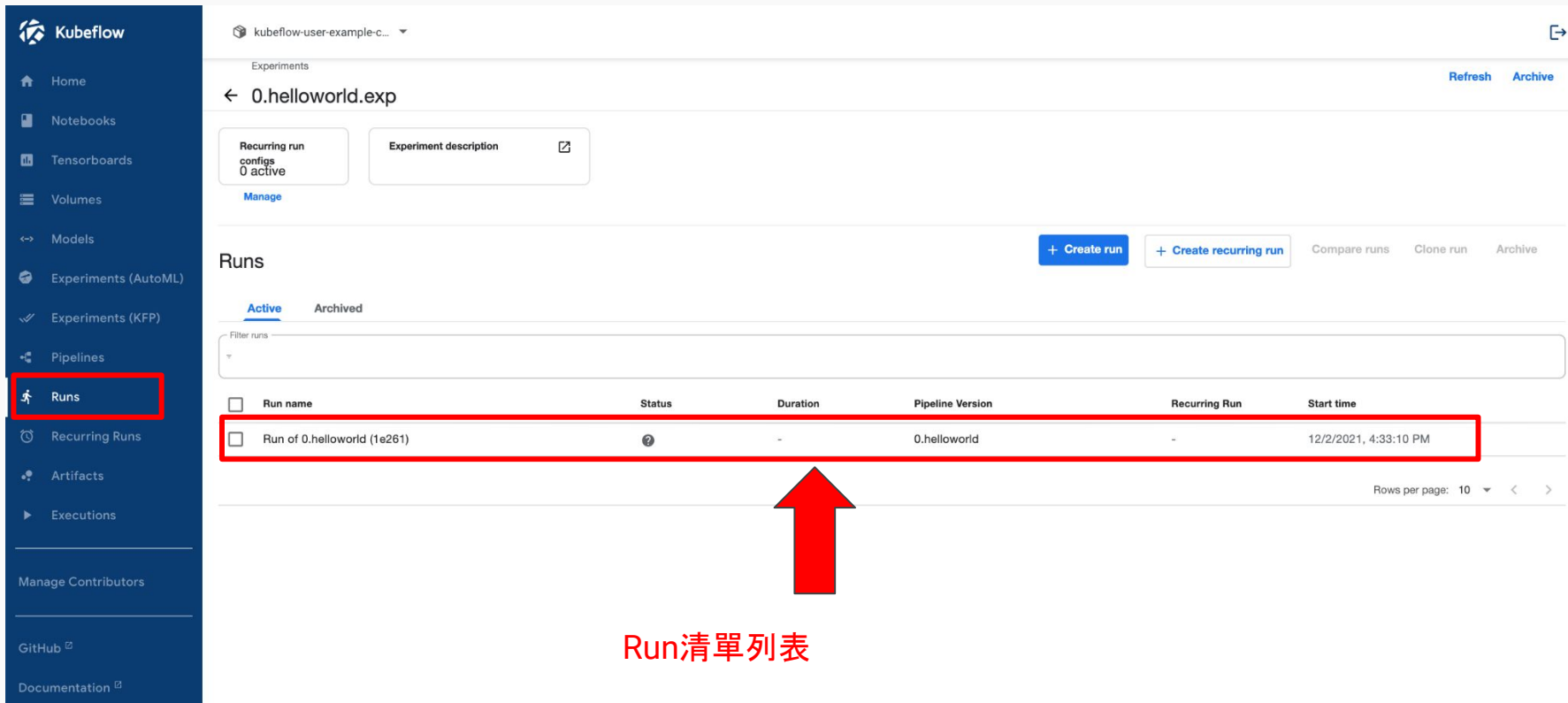
text2
message 2

[Start](#) [Skip this step](#)

1. 運行此Pipeline成為Run資源

2. 指定其歸屬的實驗資源

Step4: 建立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清單列表

Step4: 建立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 simple pipeline with two steps: 'echo1' and 'echo2', both marked with green checkmarks. 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 shows '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' (Run Result Output) to the 'main-logs' output artifact.

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

範例2 e2e 模型建立/觀察/上線

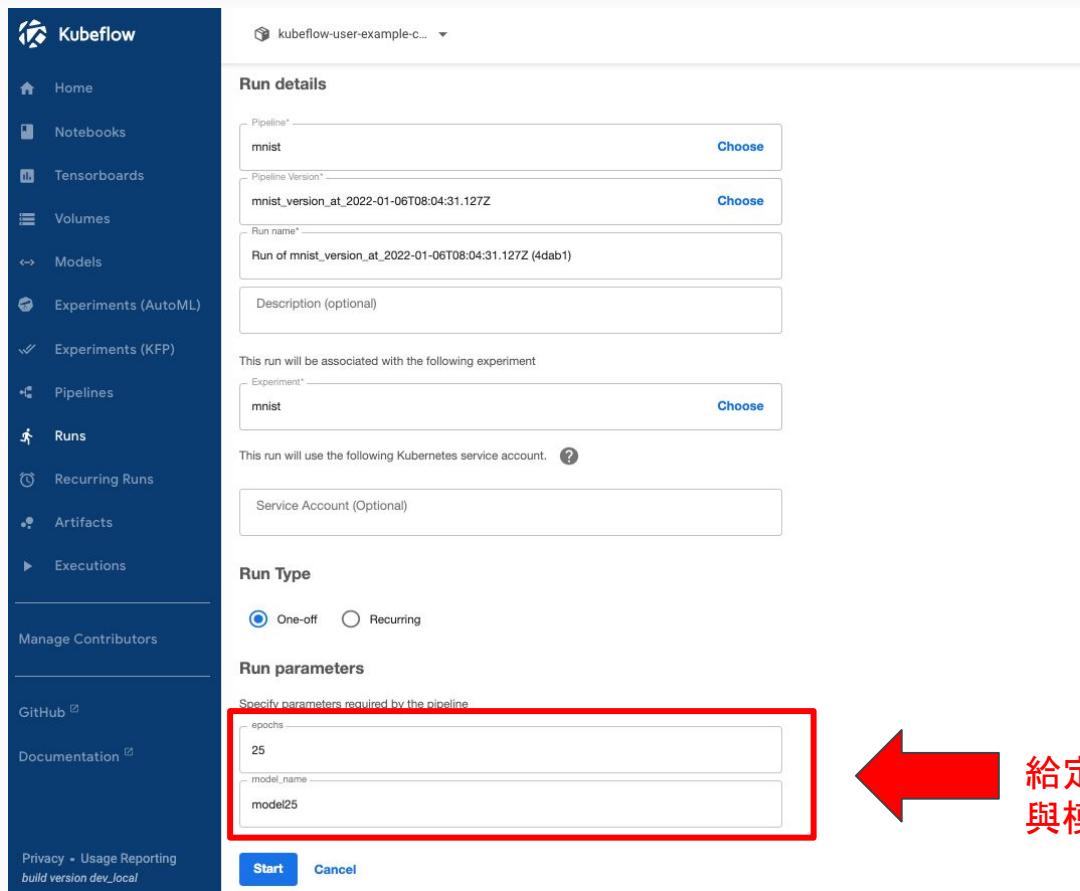
Step1: 建立Mnist模型 (1/3)

The screenshot displays a JupyterLab environment with two open notebooks: '1.build-mnist-model.ipynb' and '2.kserve.ipynb'. The left sidebar shows a file explorer with the path '/ ... / tutorials / kserve /' highlighted in a red box. The main area shows the '2.kserve.ipynb' notebook with a code cell that has been executed, displaying a 3x3 grid of handwritten digits from the MNIST dataset. The digits are labeled as Class 9, Class 0, Class 3, Class 9, Class 5, Class 6, Class 3, Class 9, and Class 8. The code cell shows the following code:

```
[6]: from typing import NamedTuple
import kfp
from kfp import dsl
```

The code cell also shows the output of the execution, which is a 3x3 grid of handwritten digits. The digits are labeled as Class 9, Class 0, Class 3, Class 9, Class 5, Class 6, Class 3, Class 9, and Class 8. The code cell also shows the output of the execution, which is a 3x3 grid of handwritten digits.

Step1: 建立Mnist模型 (2/3)



The image shows the Kubeflow web interface for creating a new run. The left sidebar contains navigation links: Home, Notebooks, Tensorboards, Volumes, Models, Experiments (AutoML), Experiments (KFP), Pipelines, Runs, Recurring Runs, Artifacts, Executions, Manage Contributors, GitHub, and Documentation. The main content area is titled 'Run details' and includes fields for Pipeline (mnist), Pipeline Version (mnist_version_at_2022-01-06T08:04:31.127Z), Run name (Run of mnist_version_at_2022-01-06T08:04:31.127Z (4dab1)), and Description (optional). Below these is a section for 'Run Type' with radio buttons for 'One-off' (selected) and 'Recurring'. The 'Run parameters' section is highlighted with a red box and contains two input fields: 'epochs' with the value '25' and 'model_name' with the value 'model25'. A red arrow points from the right towards this red box. At the bottom of the form are 'Start' and 'Cancel' buttons.

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*
mnist [Choose](#)

Pipeline Version*
mnist_version_at_2022-01-06T08:04:31.127Z [Choose](#)

Run name*
Run of mnist_version_at_2022-01-06T08:04:31.127Z (4dab1)

Description (optional)

This run will be associated with the following experiment

Experiment*
mnist [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

epochs
25

model_name
model25

[Start](#) [Cancel](#)

給定模型參數
與模型名稱

Step1: 建立Mnist模型 (3/3)

Kubeflow

Home

Notebooks

Tensorboards

Volumes

Models

Experiments (AutoML)

Experiments (KFP)

Pipelines

Runs

Recurring Runs

Artifacts

Executions

Manage Contributors

GitHub

Privacy • Usage Reporting
build version dev_local

kubeflow-user-example-c...

Experiments > mnist

Run of mnist_version_at_2022-01-06T08:04:31.127Z (7cfe8)

Retry Clone run Terminate Archive

Graph Run output Config

Simplify Graph

mypvc

Mnist func

Tensorboard func

Input/Output

Visualizations

Details

Volumes

Logs

Pod

Events

ML Metadata

Input parameters

epochs

25

model_name

model25

mypvc-name

mnist-pipeline-l892p-newpvc

Input artifacts

Output parameters

logdir

/persist-log/20220106-080616

Output artifacts

logdir

minio://mlpipeline/artifacts/mnist-pipeline-bsnjh/2022/01/06/mnist-pipeline-bsnjh-4082692353/mnist-func-logdir.tgz

View All

/persist-log/20220106-080616

main-logs

minio://mlpipeline/artifacts/mnist-pipeline-bsnjh/2022/01/06/mnist-pipeline-bsnjh-4082692353/main.log

View All


mnist_func: /persist-log

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist/11490434 [.....] - ETA: 0s 4202496/1149043

...

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

Step2: 分析Mnist模型-Tensorboard (1/5) - Optional

 **Kubeflow**

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

Manage Contributors

Privacy • Usage Reporting
build version dev_local

kubeflow-user-example-c...

Experiments > run-mnist

Run of 5.mnist_version_at_2021-12-03T03:26:49.971Z (dc2e1)

Retry Clone run Terminate Archive

Graph Run output Config

Simplify Graph

mypvc

Mnist func

Tensorboard func

Tensorboard

TF Image

TensorFlow 2.2.2

Start Tensorboard

Visualization Cre

create visuali manually

mnist-pipeline-gxj5d-4191388434

Input/Output Visualizations Details Volumes Logs Pod Events ML Metadata

建立Tensorboard作為支援觀察

Add visualizations to your own components following instructions in [Visualize Results in the Pipelines UI](#).

Runtime execution graph. Only steps that are currently running or

Step2: 分析Mnist模型-Tensorboard (2/5) - Optional

(此步為進階功能, 若無法實作也無仿)

為了讓tensorboard可以查看本地端的資料, 我們得將其對應資源建立起:

1. 先獲得viewer的id

```
root@instance-y:/# kubectl get viewer -n kubeflow-user-example-com
```

NAME	AGE
viewer-7c323b7907688f3f3fd81151c62eb58271a73803	30s

2. 先尋找pvc的位置

```
root@instance-1:~# kubectl get pvc -n kubeflow-user-example-com
```

NAME	STATUS	VOLUME	CA
volumeop-sequential-mv8sz-newpvc	Bound	pvc-17431607-83bd-44f3-abe9-79b0108ae1cc	1G
workspace-demo1	Bound	pvc-cf2d97d1-0202-4e2e-bafe-f398562627e2	5G

3. 修改至hack/tensorboard-use-local-volume.yaml底下

```
apiVersion: kubeflow.org/v1beta1
kind: Viewer
metadata:
  name: <viewer-name>
  namespace: kubeflow-user-example-com
spec:
  podTemplateSpec:
    spec:
      containers:
      - volumeMounts:
        - mountPath: /data
          name: mypvc
      serviceAccountName: default-editor
      volumes:
      - name: mypvc
        persistentVolumeClaim:
          claimName: <pvc-claim-name>
  tensorBoardSpec:
    logDir: /data
    tensorflowImage: footprintai/tensorboard:2.7.0 # see https://github.com/FootprintAI/tensorboardd
    type: tensorboard
```

Step2: 分析Mnist模型-Tensorboard (3/5) - Optional

(此步為進階功能, 若無法實作也無仿)

為了讓tensorboard可以查看本地端的資料, 我們得將其對應資源建立起:

4. 刪掉舊有Viewer並先增新的Viewer (or
`kubectl delete viewer -n kubeflow-user-example-com --all`)

```
root@instance-y:/# kubectl delete viewer viewer-7c323b7907688f3f3fd81151c62eb58271a73803 -n kubeflow-user-example-com
viewer.kubeflow.org "viewer-7c323b7907688f3f3fd81151c62eb58271a73803" deleted
```

```
root@instance-y:/# kubectl apply -f viewer.yaml
viewer.kubeflow.org/viewer-7c323b7907688f3f3fd81151c62eb58271a73803 created
```

Step2: 分析Mnist模型-Tensorboard (4/5) - Optional

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 Privacy • Usage Reporting. The main content area shows the 'Run of 5.mnist_version_at_2021-12-03T03:26:49.971Z (dc2e1)' under the 'Experiments > run-mnist' path. It includes tabs for Graph, Run output, and Config. The 'Graph' tab shows a pipeline with three steps: 'mypvc', 'Mnist func', and 'Tensorboard func', all marked with green checkmarks. A 'Simplify Graph' toggle is present. To the right, a modal window titled 'mnist-pipeline-gxj5d-4191388434' is open, showing the 'Visualizations' tab. It contains a 'Tensorboard' section with the text 'Tensorboard tensorflow/tensorflow:2.2.2 is running for this output.' and a blue button labeled 'Open Tensorboard' which is highlighted with a red rectangle. Below this is a 'Visualization Creator' section with a link 'create visualizations manually'. A large red arrow points from the Chinese text '需稍待片刻，待伺服器載入 Tensorboard 映像檔' (Need to wait a moment for the server to load the Tensorboard image) towards the 'Open Tensorboard' button.

Kubeflow

kubeflow-user-example-c...

Experiments > run-mnist

Run of 5.mnist_version_at_2021-12-03T03:26:49.971Z (dc2e1)

Graph Run output Config

Simplify Graph

mypvc

Mnist func

Tensorboard func

mnist-pipeline-gxj5d-4191388434

Input/Output Visualizations Details Volumes Logs Pod Events ML Metadata

Tensorboard

Tensorboard tensorflow/tensorflow:2.2.2 is running for this output.

Open Tensorboard

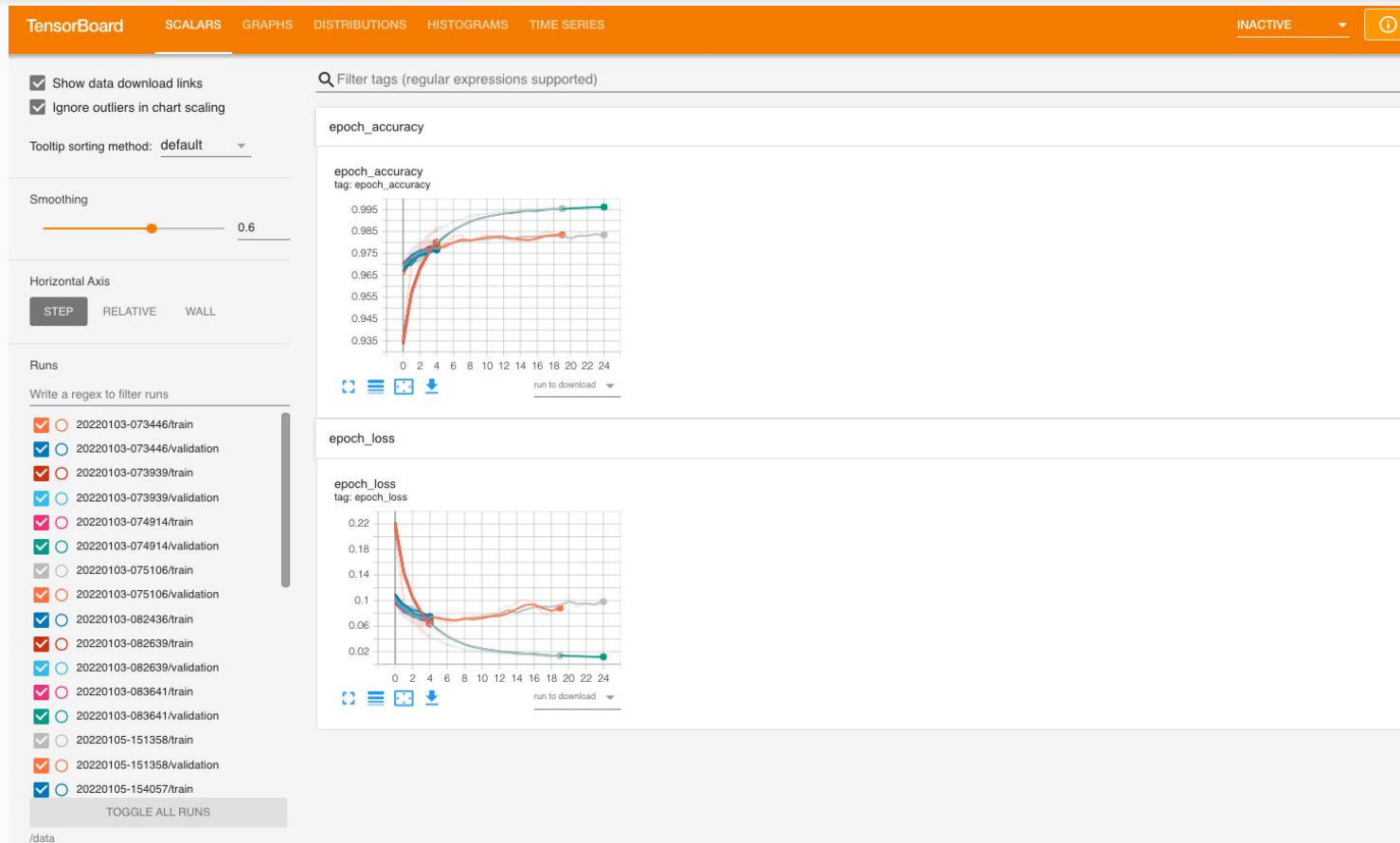
Stop Tensorboard

Visualization Creator

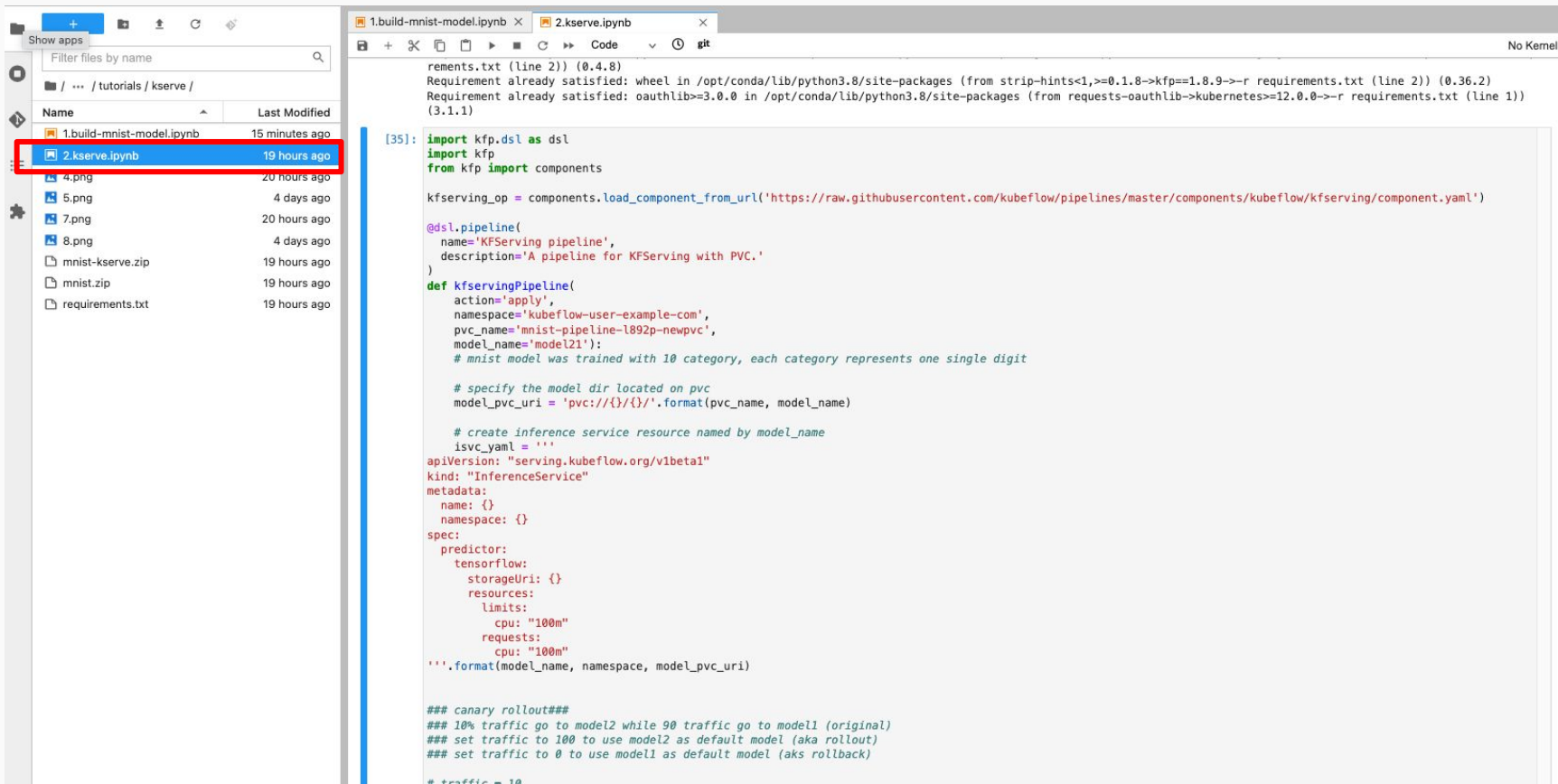
create visualizations manually

需稍待片刻，待伺服器載入 Tensorboard 映像檔

Step2: 分析Mnist模型-Tensorboard (5/5) - Optional



Step3: KServe with tensorflow model (1/4)



The screenshot shows a JupyterLab environment. On the left, a file explorer sidebar displays a directory structure under '/tutorials/kserve/'. The files listed are: '1.build-mnist-model.ipynb' (15 minutes ago), '2.kserve.ipynb' (19 hours ago, highlighted with a red box), '4.png' (20 hours ago), '5.png' (4 days ago), '7.png' (20 hours ago), '8.png' (4 days ago), 'mnist-ksvc.zip' (19 hours ago), 'mnist.zip' (19 hours ago), and 'requirements.txt' (19 hours ago). The main area shows the '2.kserve.ipynb' notebook with the following code:

```
requirements.txt (line 2)) (0.4.8)
Requirement already satisfied: wheel in /opt/conda/lib/python3.8/site-packages (from strip-hints<1,>=0.1.8->kfp==1.8.9->r requirements.txt (line 2)) (0.36.2)
Requirement already satisfied: oauthlib>=3.0.0 in /opt/conda/lib/python3.8/site-packages (from requests-oauthlib->kubernetes>=12.0.0->r requirements.txt (line 1)) (3.1.1)

[35]: import kfp.dsl as dsl
import kfp
from kfp import components

kfserving_op = components.load_component_from_url('https://raw.githubusercontent.com/kubeflow/pipelines/master/components/kubeflow/kfserving/component.yaml')

@dsl.pipeline(
    name='KFServing pipeline',
    description='A pipeline for KFServing with PVC.'
)
def kfservingPipeline(
    action='apply',
    namespace='kubeflow-user-example-com',
    pvc_name='mnist-pipeline-l892p-newpvc',
    model_name='model121'):
    # mnist model was trained with 10 category, each category represents one single digit

    # specify the model dir located on pvc
    model_pvc_uri = 'pvc://{}/{}'.format(pvc_name, model_name)


    # create inference service resource named by model_name
    isvc_yaml = '''
apiVersion: "serving.kubeflow.org/v1beta1"
kind: "InferenceService"
metadata:
  name: {}
  namespace: {}
spec:
  predictor:
    tensorflow:
      storageUri: {}
      resources:
        limits:
          cpu: "100m"
        requests:
          cpu: "100m"
    '''

    return kfserving_op(model_name, namespace, model_pvc_uri)

### canary rollout###
### 10% traffic go to model2 while 90 traffic go to model1 (original)
### set traffic to 100 to use model2 as default model (aka rollout)
### set traffic to 0 to use model1 as default model (aks rollback)

# traffic = 10
```

Step3: KServe with tensorflow model (2/4)

 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*

mnist-serve

Choose

Pipeline Version*

mnist-serve_version_at_2022-01-06T08:27:42.083Z

Choose

Run name*

Run of mnist-serve_version_at_2022-01-06T08:27:42.083Z (12d9f)

Description (optional)

This run will be associated with the following experiment

Experiment*

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

action

apply

namespace

kubeflow-user-example-com

pvc_name

mnist-pipeline-l892p-newpvc

model_name

model21

Start

Cancel

給定pvc位置與先前生成之模型名稱

Step3: KServe with tensorflow model (3/4)

獲得用戶權杖:

View -> Developer -> Developer Tools -> Application

The screenshot shows the Kubeflow console interface. On the left is a sidebar with 'Kubeflow' logo and navigation links: Home, Notebooks, Privacy, Usage Reporting, and build version dev_local. The main area displays 'Run of tf-serving (827a3)' with tabs for Graph, Run output, and Config. Below this is a table of cookies. The 'authservice_session' cookie is highlighted with a red box. The table has columns: Name, Value, Domain, Path, Expires, Size, HttpOnly, Secure, SameSite, and Priority.

Name	Value	Do...	P.	Expire...	Size	HttpOnly	Secure	Same...	Same...	Priority
1P_JAR	2021-12-02-05	loc...	/	2022-...	19		✓	None		Medium
authservice_session	MTYzODQzMjc4OXxOd3dBTRkRkVVZsZE5RMFV6TlU5YVREWkxWa1JKUVZ...	loc...	/	2021-...	179					Medium
_yaf	2021-12-02-05	loc...	/	2021-...	59					Medium

Step3: KServe with tensorflow model (4/4)

更新模型名稱

更新用戶權杖

模型預測結果

我們剛做了...

- 實現遠端開發, 訓練, 以及部署之能力
- 流水線模型開發模式
- 分散式訓練
- 模型檢測與分析
- 模型部署

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 Big Apple Wheel visible on the right side of the image.

Thank You
Questions?

Contact Us:
partnership@footprint-ai.com

- Documentations
 - <https://www.kubeflow.org/>
- Kubectl cheatsheet
 - <https://kubernetes.io/docs/reference/kubectl/cheatsheet/>