部署應用程式
NetApp Solutions

NetApp
April 12, 2024

This PDF was generated from https://docs.netapp.com/zh-tw/netapp-solutions/ai/mlrun_get_code_from_github.html on April 12, 2024. Always check docs.netapp.com for the latest.

目錄

部署應用程式 1
從GitHub取得程式碼 1
設定工作環境 1
部署Grafana儀表板 13

部署應用程式

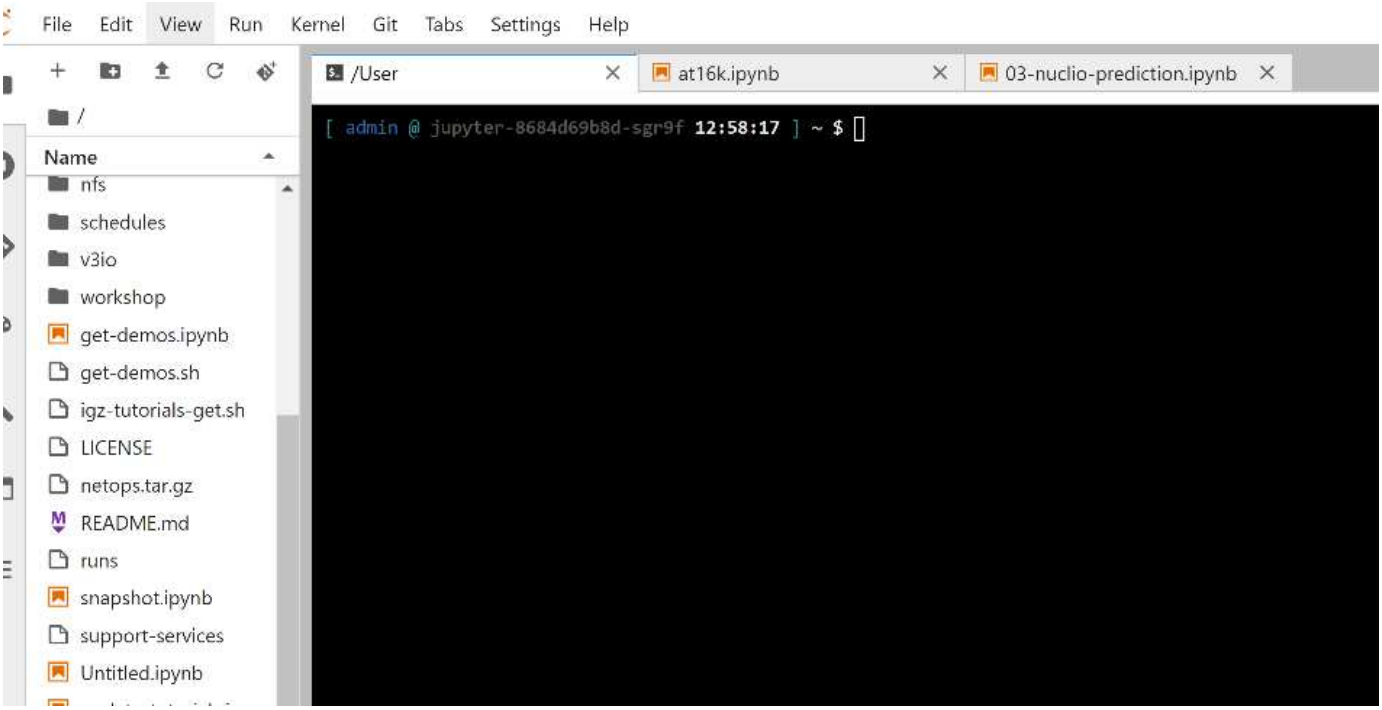
下列各節將說明如何安裝及部署應用程式。

從GitHub取得程式碼

現在NetApp Cloud Volume或NetApp Trident Volume可用於Iguazio叢集和開發人員環境、您就可以開始檢閱應用程式。

使用者擁有自己的工作區（目錄）。在每個筆記型電腦上、使用者目錄的路徑都是「/使用者」。Iguazio平台負責管理目錄。如果您依照上述指示操作、NetApp Cloud Volume可在「/NetApp」目錄中找到。

使用Jupyter終端從GitHub取得程式碼。



在Jupyter終端機提示字元下、複製專案。

```
cd /User
git clone .
```

現在您應該可以在Jupyter工作區的檔案樹狀結構中看到「netops-netapp'」資料夾。

設定工作環境

將「Notebook」（筆記型電腦）的「set_env-example.ipynb」複製為「set_env.ipynb」。開啟並編輯「set_env.ipynb」。此筆記型電腦可設定認證資料、檔案位置及執行驅動程式的變數。

如果您依照上述指示進行、下列步驟是您唯一要做的變更：

1. 從Iguazio服務儀表板取得此值：「Docker_registry」

   範例：「Docker-registry.default-tenant.app.clusterq.iguaziodev.com:80`

2. 將「admin」變更為您的Iguazio使用者名稱：

   「IGZ_container路徑='/user/admin'」

   以下ONTAP 是有關「系統連線」的詳細資料。包括安裝Trident時所產生的Volume名稱。下列設定適用於內部部署ONTAP 的內部部署的叢集：

```
ontapClusterMgmtHostname = '0.0.0.0'
ontapClusterAdminUsername = 'USER'
ontapClusterAdminPassword = 'PASSWORD'
sourceVolumeName = 'SOURCE VOLUME'
```

   下列設定適用於Cloud Volumes ONTAP 下列項目：

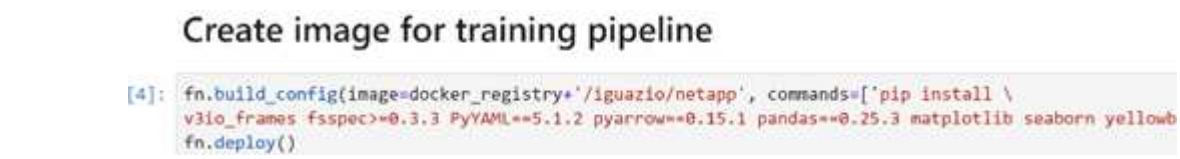
```
MANAGER=ontapClusterMgmtHostname
svm='svm'
email='email'
password=ontapClusterAdminPassword
weid="weid"
volume=sourceVolumeName
```

建立基礎Docker映像檔

Iguazio平台包含建置ML管線所需的一切。開發人員可以定義執行管線所需的Docker映像規格、以及從Jupyter Notebook建立映像。開啟筆記型電腦「create- images.ipynb」、然後執行「All」儲存格。

這款筆記型電腦會建立兩個我們在銷售管道中使用的映像。

- 《iguazio/NetApp》 用於處理ML工作。



- 「NetApp/Pipeline」。包含處理NetApp Snapshot複本的公用程式。

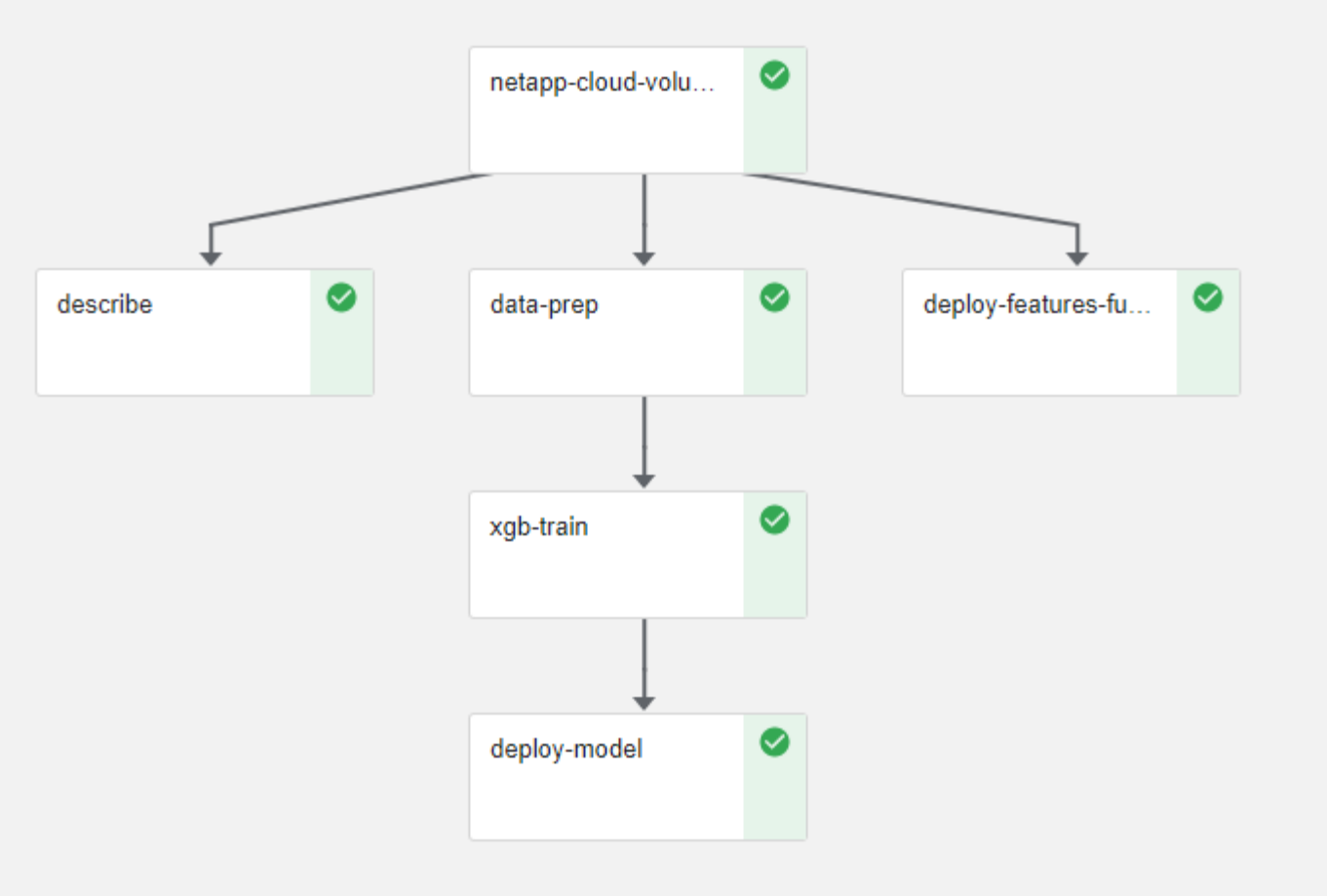
檢閱個別的**Jupyter**筆記型電腦

下表列出我們用來建置此工作的程式庫和架構。所有這些元件都已與Iguazio的角色型存取與安全控制功能完全整合。

程式庫/架構	說明
MLRun	由Iguazio管理、可讓您組裝、執行及監控ML/AI傳輸途徑。
Nuclio	與Iguazio整合的無伺服器功能架構。也可作為由Iguazio管理的開放原始碼專案。
Kubeflow	以Kubernetes為基礎的架構、用於部署管線。這也是Iguazio所貢獻的開放原始碼專案。它與Iguazio整合、可提升安全性、並與其他基礎架構整合。
Docker	Docker登錄是以服務形式在Iguazio平台上執行。您也可以變更此設定以連線至登錄。
NetApp Cloud Volumes	在AWS上執行的Cloud Volumes可讓我們存取大量資料、並可將Snapshot複本複製到訓練所用的資料集版本。
Trident	Trident是由NetApp管理的開放原始碼專案。它有助於整合Kubernetes的儲存與運算資源。

我們使用數個筆記型電腦來建構ML管線。每一部筆記型電腦都能在整合到銷售管道之前進行個別測試。我們會依照此示範應用程式的部署流程、分別說明每一部筆記型電腦。

所需的結果是一條管道、會根據資料的Snapshot複本來訓練模型、並部署模型以供參考。下圖顯示完整MLRun管線的區塊圖。



部署資料產生功能

本節說明我們如何使用Nuclio無伺服器功能來產生網路裝置資料。使用案例可從部署管線的Iguazio用戶端進行調整、並使用Iguazio服務來監控和預測網路裝置故障。

我們模擬來自網路裝置的資料。執行Jupyter筆記型電腦「data- gener.ipynb」可建立無伺服器功能、每10分鐘執行一次、並產生含有新資料的Parquet檔案。若要部署此功能、請執行此筆記型電腦中的所有儲存格。請參閱 "Nuclio網站" 檢閱此筆記型電腦中任何不熟悉的元件。

產生函數時會忽略具有下列註解的儲存格。筆記型電腦中的每個儲存格都會被視為功能的一部分。匯入Nuclio模組以啟用「%nuclio fic」。

```
# nuclio: ignore
import nuclio
```

在該函數的規格中、我們定義了執行該函數的環境、觸發方式、以及它所耗用的資源。

```
spec = nuclio.ConfigSpec(config={"spec.triggers.inference.kind":"cron",

"spec.triggers.inference.attributes.interval" :"10m",
                                  "spec.readinessTimeoutSeconds" : 60,
                                  "spec.minReplicas" : 1},……
```

初始化功能時、Nuclio架構會叫用「init_context」功能。

```
def init_context(context):
    ….
```

當函數初始化時、會叫用任何不在函數中的程式碼。當您叫用它時、會執行處理常式功能。您可以變更處理常式的名稱、並在函數規格中加以指定。

```
def handler(context, event):
            …
```

您可以在部署之前、先從筆記型電腦測試功能。

```
%%time
# nuclio: ignore
init_context(context)
event = nuclio.Event(body='')
output = handler(context, event)
output
```

此功能可從筆記型電腦部署、也可從CI/CD管道部署（修改此程式碼）。

```
addr = nuclio.deploy_file(name='generator',project='netops',spec=spec,
tag='v1.1')
```

管道筆記型電腦

這些筆記型電腦不應個別執行此設定。這只是對每個筆記型電腦的審查。我們將它們視為管道的一部分。若要個別執行、請檢閱MLRun文件、以Kubernetes工作的形式執行。

Snap_CV.ipynb

此筆記型電腦會在管線開始時處理Cloud Volume Snapshot複本。它會將磁碟區名稱傳遞給管線內容。此筆記型電腦會叫用Shell指令碼來處理Snapshot複本。在管線中執行時、執行內容會包含變數、以協助找出執行所需的所有檔案。撰寫此程式碼時、開發人員不必擔心執行程式碼的容器中的檔案位置。如稍後所述、此應用程式會隨其所有相依性一起部署、而且是提供執行內容的管線參數定義。

```
command = os.path.join(context.get_param('APP_DIR'),"snap_cv.sh")
```

建立的Snapshot複本位置會放置在MLRun內容中、供管線中的步驟使用。

```
context.log_result('snapVolumeDetails',snap_path)
```

接下來的三部筆記型電腦會平行執行。

資料準備：**ipynb**

原始指標必須轉變為功能、才能進行模型訓練。此筆記型電腦會從Snapshot目錄讀取原始指標、並將模型訓練功能寫入NetApp Volume。

在管線內容中執行時、輸入「DAAT_DIR」會包含Snapshot複本位置。

```
metrics_table = os.path.join(str(mlruncontext.get_input('DATA_DIR',
os.getenv('DATA_DIR','/netpp'))),
                             mlruncontext.get_param('metrics_table',
os.getenv('metrics_table','netops_metrics_parquet')))
```

描述**.ipynb**

為了視覺化傳入的度量、我們部署了一個管線步驟、提供可透過Kubeflow和MLRun UI取得的繪圖和圖表。每次執行都有其專屬版本的視覺化工具。

```
ax.set_title("features correlation")
plt.savefig(os.path.join(base_path, "plots/corr.png"))
context.log_artifact(PlotArtifact("correlation",  body=plt.gcf()),
local_path="plots/corr.html")
```

Deploy功能**.ipynb**

我們持續監控指標、以尋找異常狀況。這款筆記型電腦會建立一個無伺服器功能、產生在傳入度量上執行預測所需的功能。此筆記型電腦會啟動功能的建立。功能代碼位於筆記型電腦「data- prep.ipynb」中。請注意、我們使用同一部筆記型電腦做為此目的的管道步驟。

訓練**.ipynb**

建立這些功能之後、我們便開始進行模型訓練。此步驟的輸出是用於推斷的模型。我們也會收集統計資料、以追蹤每次執行（實驗）。

例如、下列命令會在該實驗的內容中輸入準確度分數。此值可在Kubeflow和MLRun中看到。

```
context.log_result('accuracy',score)
```

deploy推論函數.ipynb

管道的最後一步是將模型部署為無伺服器功能、以便持續推斷。此筆記型電腦會啟動建立在「nuclio-inertere-fuite.ipynb」中定義的無伺服器功能。

審查及建置管道

在管線中執行所有的筆記型電腦、可持續執行實驗、根據新的指標來重新評估模型的準確度。首先、開啟「pipe.ipynb」筆記型電腦。我們將帶您詳細瞭解NetApp與Iguazio如何簡化這項ML管線的部署。

我們使用MLRun為管線的每個步驟提供背景資料並處理資源分配。MLRun API服務在Iguazio平台上執行、是與Kubernetes資源互動的點。每個開發人員都無法直接要求資源；API會處理要求並啟用存取控制。

```
# MLRun API connection definition
mlconf.dbpath = 'http://mlrun-api:8080'
```

該管道可與NetApp Cloud Volumes和內部部署Volume搭配使用。我們打造此示範影片來使用Cloud Volumes、但您可以在程式碼中看到可在內部部署執行的選項。

```
# Initialize the NetApp snap fucntion once for all functions in a notebook
if [ NETAPP_CLOUD_VOLUME ]:
    snapfn =
code_to_function('snap',project='NetApp',kind='job',filename="snap_cv.ipyn
b").apply(mount_v3io())
    snap_params = {
    "metrics_table" : metrics_table,
    "NETAPP_MOUNT_PATH" : NETAPP_MOUNT_PATH,
    'MANAGER' : MANAGER,
    'svm' : svm,
    'email': email,
    'password': password ,
    'weid': weid,
    'volume': volume,
    "APP_DIR" : APP_DIR
       }
else:
    snapfn =
code_to_function('snap',project='NetApp',kind='job',filename="snapshot.ipy
nb").apply(mount_v3io())
….
snapfn.spec.image = docker_registry + '/netapp/pipeline:latest'
snapfn.spec.volume_mounts =
[snapfn.spec.volume_mounts[0],netapp_volume_mounts]
      snapfn.spec.volumes = [ snapfn.spec.volumes[0],netapp_volumes]
```

將Jupyter筆記型電腦轉變成Kubeflow步驟所需的第一個行動、就是將程式碼變成功能。某項功能具備執行該筆記型電腦所需的所有規格。當您向下捲動筆記本時、您會看到我們為管道中的每個步驟定義了功能。

筆記型電腦的一部分	說明
<code_to功能>（MLRun模組的一部分）	功能名稱：專案名稱。用於組織所有專案成品。這可在MLRun UI中看到。種類。在此案例中、Kubernetes工作。這可能是dask、MPI、走勢8等等。如需詳細資訊、請參閱MLRun文件。檔案：筆記型電腦的名稱。這也可以是Git（HTTP）中的位置。
映像	我們在這個步驟中使用的Docker映像檔名稱。我們先前使用cree-image.ipynb系列筆記型電腦來建立這個應用程式。
Volume_掛 載與磁碟區	執行時掛載NetApp Cloud Volume的詳細資料。

我們也定義步驟的參數。

```
params={    "FEATURES_TABLE":FEATURES_TABLE,
            "SAVE_TO" : SAVE_TO,
            "metrics_table" : metrics_table,
            'FROM_TSDB': 0,
            'PREDICTIONS_TABLE': PREDICTIONS_TABLE,
            'TRAIN_ON_LAST': '1d',
            'TRAIN_SIZE':0.7,
            'NUMBER_OF_SHARDS' : 4,
            'MODEL_FILENAME' : 'netops.v3.model.pickle',
            'APP_DIR' : APP_DIR,
            'FUNCTION_NAME' : 'netops-inference',
            'PROJECT_NAME' : 'netops',
            'NETAPP_SIM' : NETAPP_SIM,
            'NETAPP_MOUNT_PATH': NETAPP_MOUNT_PATH,
            'NETAPP_PVC_CLAIM' : NETAPP_PVC_CLAIM,
            'IGZ_CONTAINER_PATH' : IGZ_CONTAINER_PATH,
            'IGZ_MOUNT_PATH' : IGZ_MOUNT_PATH
             }
```

為所有步驟定義功能之後、您就可以建構管線。我們使用「kfp"模組來定義這個定義。使用MLRun與自行建置的差異在於編碼的簡化與縮短。

我們定義的功能會使用MLRun的「AS步驟」功能、變成步驟元件。

Snapshot步驟定義

啟動Snapshot功能、輸出及掛載v3io作為來源：

```
snap = snapfn.as_step(NewTask(handler='handler',params=snap_params),
name='NetApp_Cloud_Volume_Snapshot',outputs=['snapVolumeDetails','training
_parquet_file']).apply(mount_v3io())
```

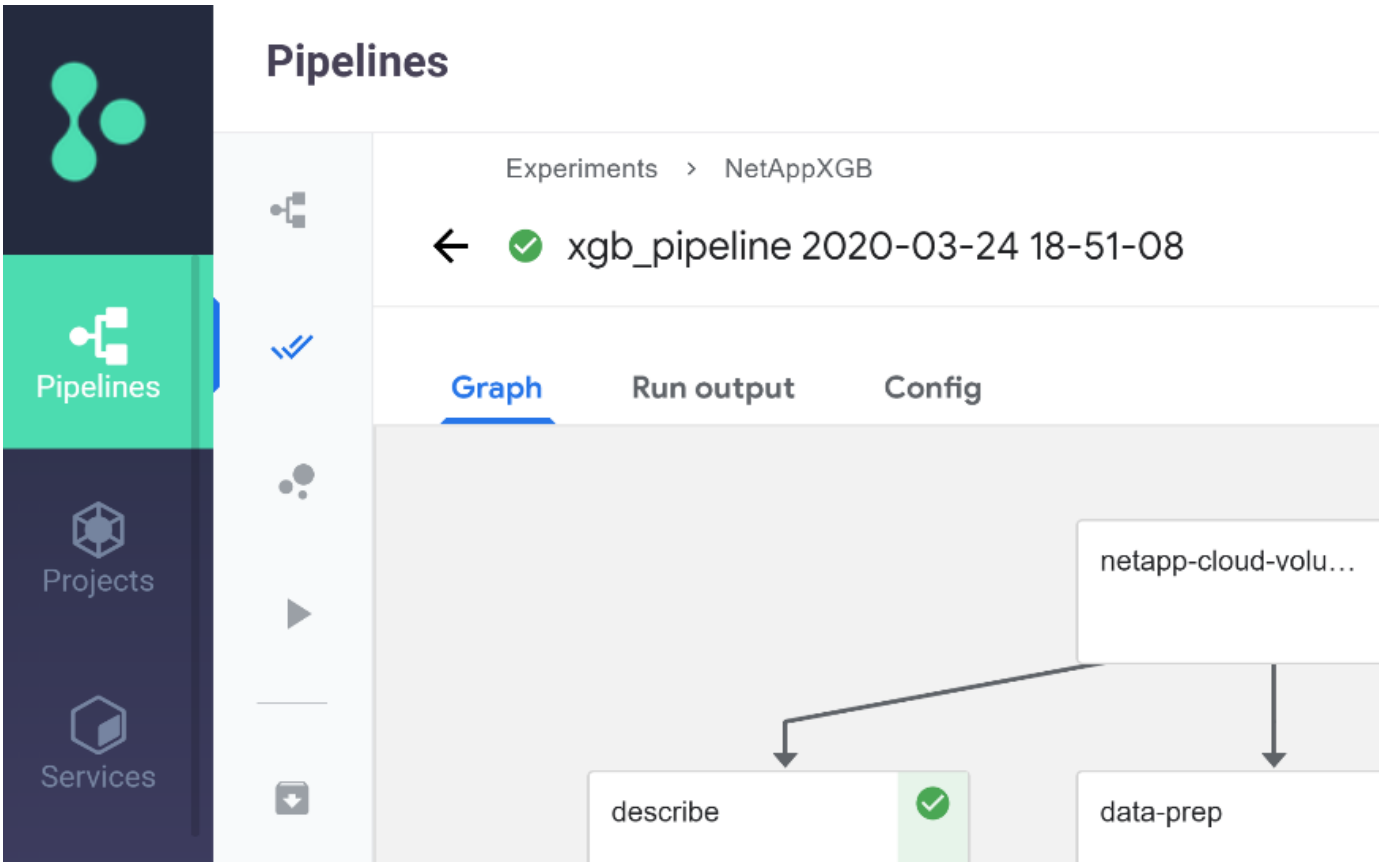
參數	詳細資料
新工作	newtask是函數執行的定義。
（MLRun模組）	處理常式：要叫用的Python函數名稱。我們在筆記型電腦中使用名稱處理常式、但這不是必要的。參數。傳遞給執行的參數。在程式碼中、我們使用context.Get_param（「參數」）來取得值。
AS步驟	名稱。Kubeflow管道步驟名稱。輸出。這些是步驟在完成時新增至字典的值。請參閱Snap_CV.ipynb系列筆記型電腦。mount_v3io()。這會設定執行管線之使用者要掛載/User的步驟。

```
prep = data_prep.as_step(name='data-prep',
handler='handler',params=params,
                          inputs = {'DATA_DIR':
snap.outputs['snapVolumeDetails']} ,

out_path=artifacts_path).apply(mount_v3io()).after(snap)
```

參數	詳細資料
輸入	您可以將前一個步驟的輸出傳送到一個步驟。在這種情況下、snap.outputs [snapVolume Details]是我們在Snapshot步驟上建立的Snapshot複本名稱。
Out_path	放置使用MLRun模組log_act件 產生成品的位置。

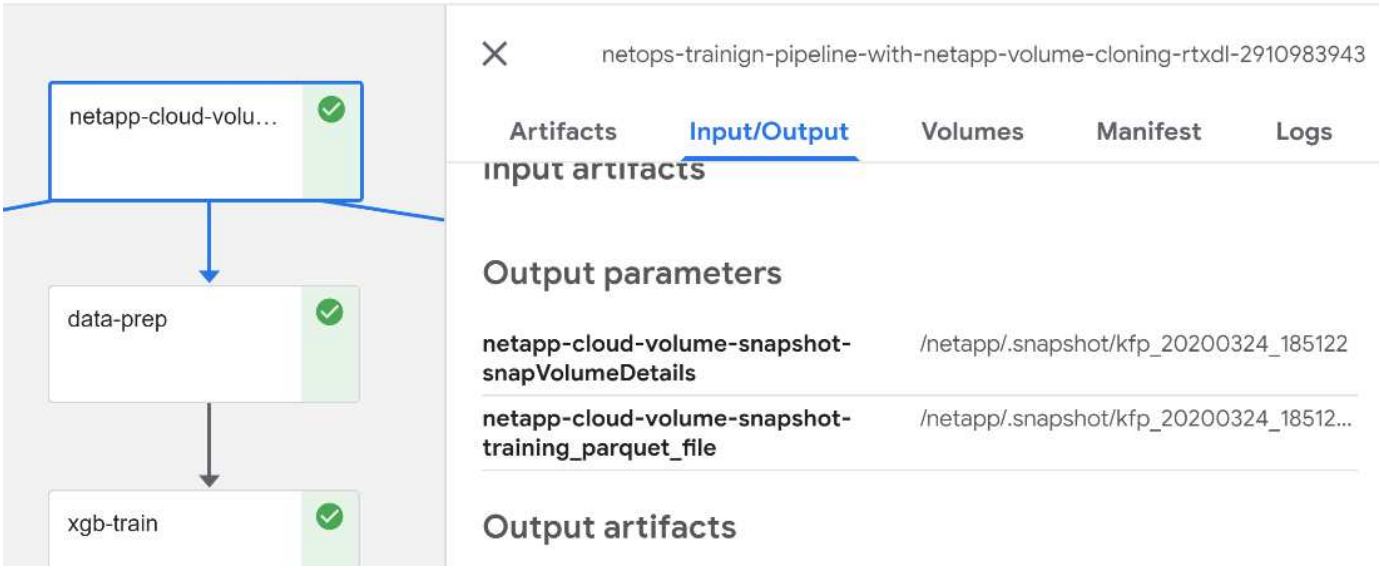
您可以從上到下執行「pipele.ipynb」。然後、您可以從Iguazio儀表板前往「Pipines」（管路）索引標籤、以監控Iguazio儀表板「Pipines」（管路）索引標籤中顯示的進度。



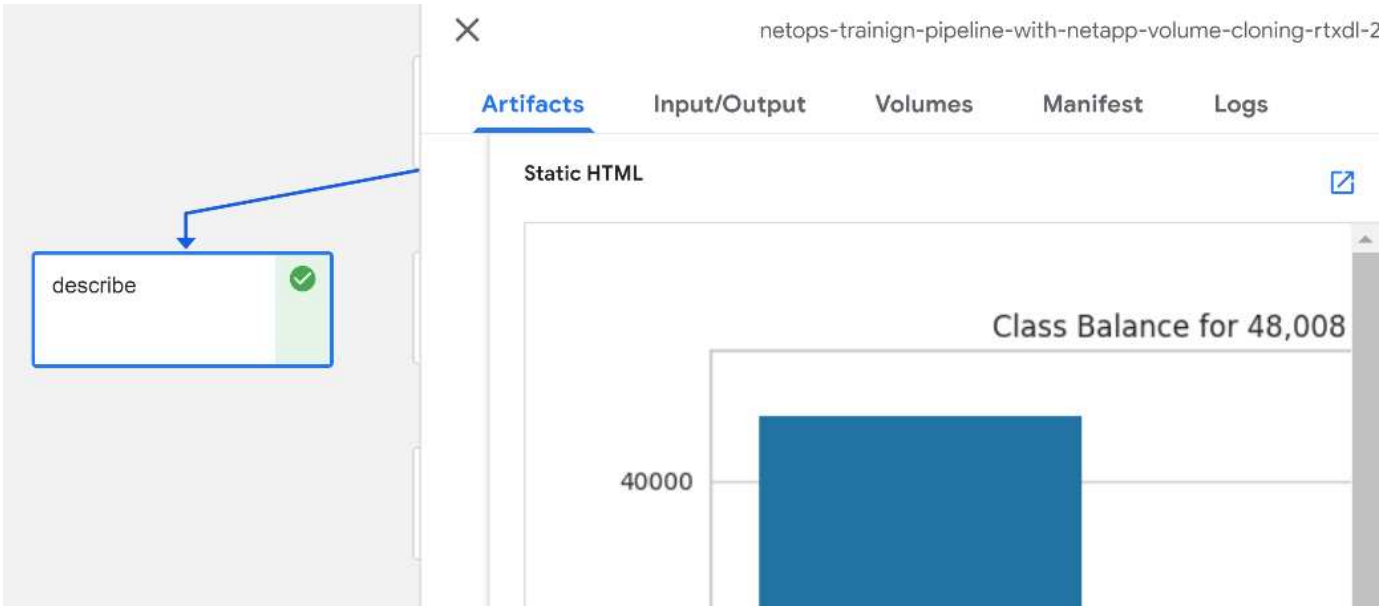
由於我們在每次路跑中都記錄訓練步驟的準確度、因此每次實驗都有精準度的記錄、如訓練準確度記錄所示。

Run name	Status	Duration	Pipeline Version	Recurring ...	Start time	accuracy
xgb_pipeline 2020-03-24 18-51-...	✓	0:08:43	[View pipeline]	-	3/24/2020, 2:51:09 PM	0.985
xgb_pipeline 2020-03-19 13-31-...	✓	0:08:14	[View pipeline]	-	3/19/2020, 9:31:19 AM	0.980
xgb_pipeline 2020-03-18 12-56-...	✓	0:08:11	[View pipeline]	-	3/18/2020, 8:56:08 AM	0.990
xgb_pipeline 2020-03-17 19-49-...	✓	0:08:03	[View pipeline]	-	3/17/2020, 3:49:31 PM	0.985
xgb_pipeline 2020-03-17 18-34-...	✓	0:05:54	[View pipeline]	-	3/17/2020, 2:34:56 PM	0.980
xgb_pipeline 2020-03-17 17-34-...	✓	0:04:48	[View pipeline]	-	3/17/2020, 1:34:16 PM	0.982
xgb_pipeline 2020-03-17 17-01-...	✓	0:05:25	[View pipeline]	-	3/17/2020, 1:01:58 PM	0.987
xgb_pipeline 2020-03-16 16-47-...	✓	0:06:08	[View pipeline]	-	3/16/2020, 12:47:19 ...	0.983
xgb_pipeline 2020-03-16 13-57-...	✓	0:05:18	[View pipeline]	-	3/16/2020, 9:57:03 AM	0.980

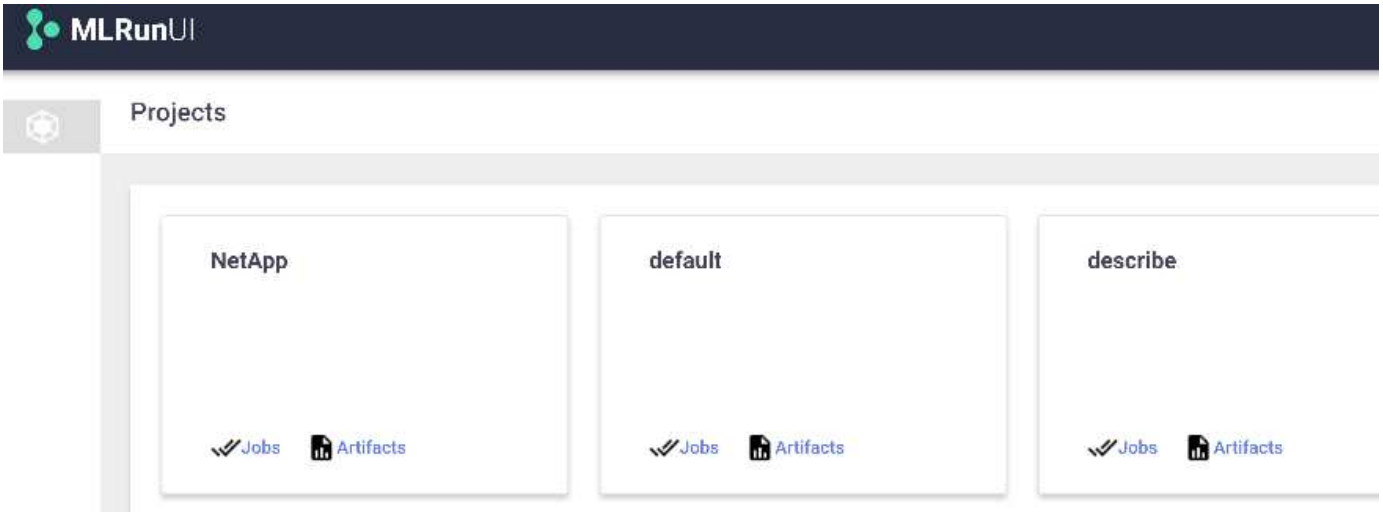
如果您選取Snapshot步驟、就會看到用於執行此實驗的Snapshot複本名稱。



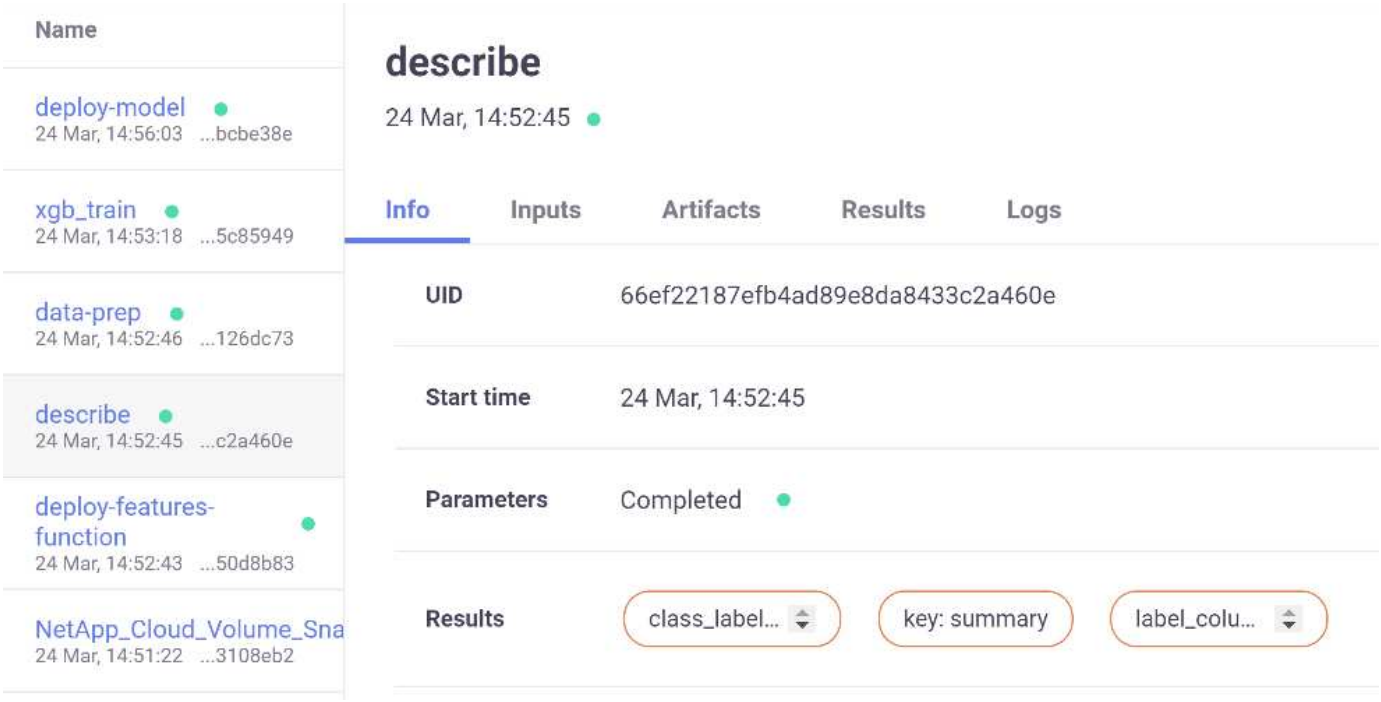
所述步驟具有視覺成品、可用來探索我們使用的指標。您可以展開以檢視完整繪圖、如下圖所示。



MLRun API資料庫也會追蹤各專案所組織之每個執行的輸入、輸出和成品。下列影像提供每個掃描的輸入、輸出和成品範例。



我們會針對每項工作儲存其他詳細資料。



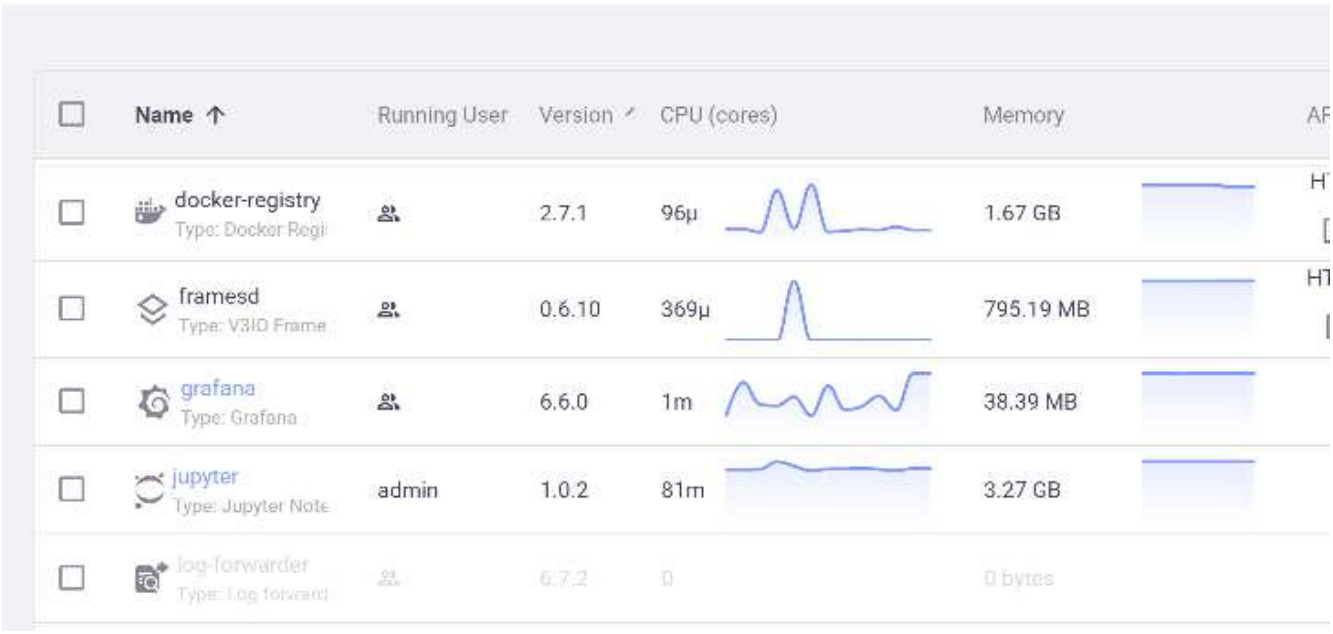
關於MLRun的資訊比本文件所涵蓋的資訊更多。所有成品（包括步驟和功能的定義）都可儲存至API資料庫、版本控制、個別或完整的專案來叫用。專案也可儲存並推送至Git供日後使用。我們鼓勵您在上深入瞭解 "MLRun GitHub網站"。

部署Grafana儀表板

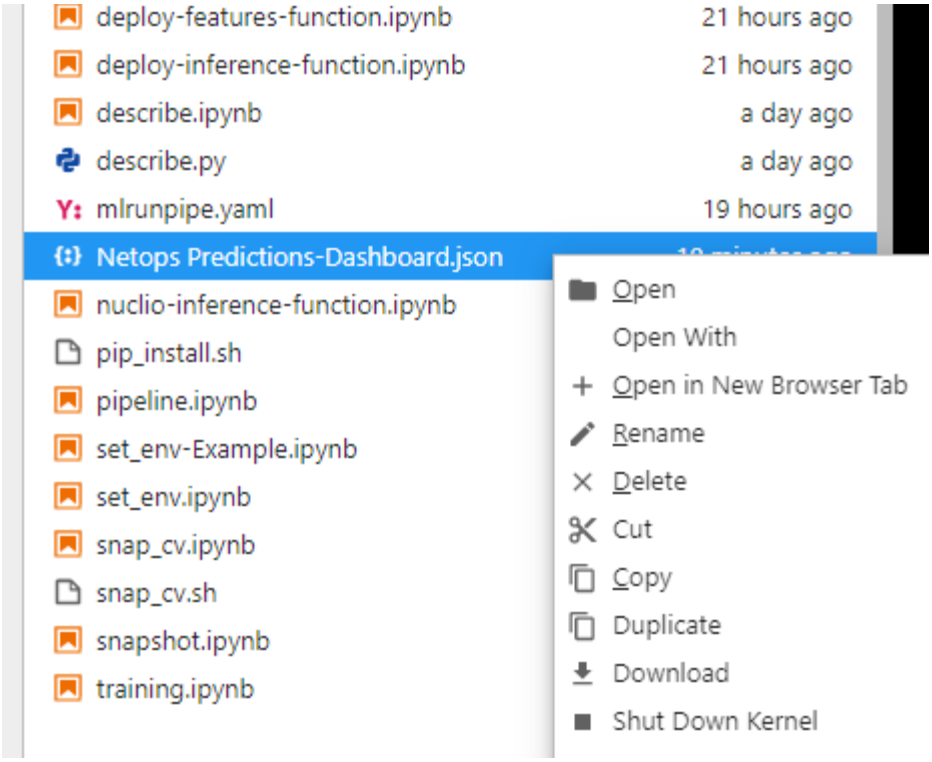
部署所有項目之後、我們會針對新資料執行推斷。模型預測網路裝置設備故障。預測結果會儲存在Iguazio Timezio表格中。您可以在整合了Iguazio安全性與資料存取原則的平台上、以Grafana視覺化的方式呈現結果。

您可以將所提供的Json檔案匯入叢集中的Grafana介面、以部署儀表板。

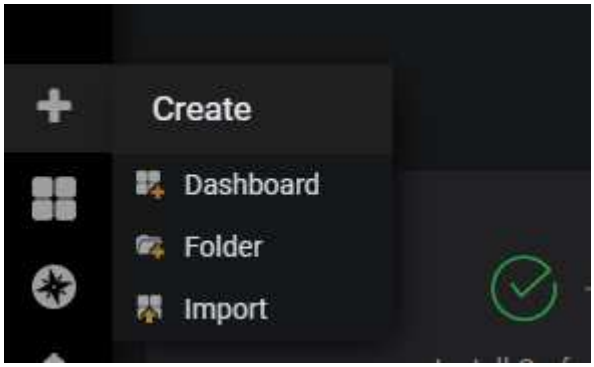
1. 若要驗證Grafana服務是否正在執行、請查看「服務」下的。



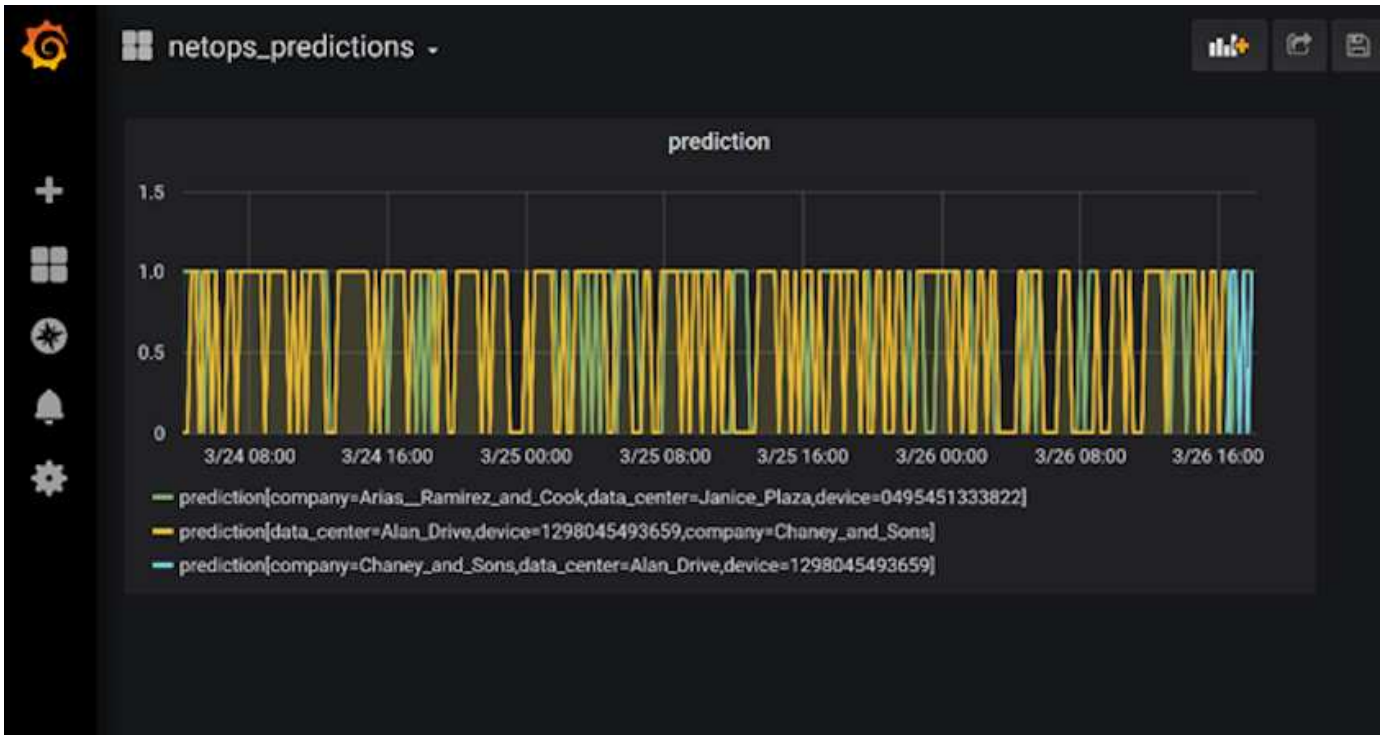
2. 如果不存在、請從「服務」區段部署執行個體：
   a. 按一下「New Service（新服務
   b. 從清單中選取Grafana。
   c. 接受預設值。
   d. 按「Next Step（下一步）」
   e. 輸入您的使用者ID。
   f. 按一下儲存服務。
   g. 按一下頂端的套用變更。
3. 若要部署儀表板、請透過Jupyter介面下載「NetopsPredictions-Dashboard・json」檔案。



4. 從「服務」區段開啟Grafana、然後匯入儀表板。



5. 按一下「上傳」「*。json」檔案、然後選取您先前下載的檔案（「NetopsPredictions-Dashboard・json」）。儀表板會在上傳完成後顯示。



部署清理功能

當您產生大量資料時、務必保持一切乾淨且有組織。若要這麼做、請使用「cleanup.ipynb」筆記型電腦來部署清理功能。

效益

NetApp和Iguazio藉由在Kubeflow、Apache Spark和TensorFlow等重要架構中建置、以及Docker和Kubernetes等協調工具、來加速和簡化AI和ML應用程式的部署。NetApp與Iguazio藉由統一化端點對端點資料傳輸途徑、有效縮短許多進階運算工作負載固有的延遲與複雜度、有效縮短開發與營運之間的落差。資料科學家可以在大型資料集上執行查詢、並在訓練階段與授權使用者安全地共用資料與演算法模型。在容器化模型準備就緒可供正式作業之後、您可以輕鬆地將其從開發環境移至作業環境。

版權資訊

Copyright © 2024 NetApp, Inc. 版權所有。台灣印製。非經版權所有人事先書面同意，不得將本受版權保護文件的任何部分以任何形式或任何方法（圖形、電子或機械）重製，包括影印、錄影、錄音或儲存至電子檢索系統中。

由 NetApp 版權資料衍伸之軟體必須遵守下列授權和免責聲明：

此軟體以 NETAPP「原樣」提供，不含任何明示或暗示的擔保，包括但不限於有關適售性或特定目的適用性之擔保，特此聲明。於任何情況下，就任何已造成或基於任何理論上責任之直接性、間接性、附隨性、特殊性、懲罰性或衍生性損害（包括但不限於替代商品或服務之採購；使用、資料或利潤上的損失；或企業營運中斷），無論是在使用此軟體時以任何方式所產生的契約、嚴格責任或侵權行為（包括疏忽或其他）等方面，NetApp 概不負責，即使已被告知有前述損害存在之可能性亦然。

NetApp 保留隨時變更本文所述之任何產品的權利，恕不另行通知。NetApp 不承擔因使用本文所述之產品而產生的責任或義務，除非明確經過 NetApp 書面同意。使用或購買此產品並不會在依據任何專利權、商標權或任何其他 NetApp 智慧財產權的情況下轉讓授權。

本手冊所述之產品受到一項（含）以上的美國專利、國外專利或申請中專利所保障。

有限權利說明：政府機關的使用、複製或公開揭露須受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中的「技術資料權利 - 非商業項目」條款 (b)(3) 小段所述之限制。

此處所含屬於商業產品和 / 或商業服務（如 FAR 2.101 所定義）的資料均為 NetApp, Inc. 所有。根據本協議提供的所有 NetApp 技術資料和電腦軟體皆屬於商業性質，並且完全由私人出資開發。 美國政府對於該資料具有非專屬、非轉讓、非轉授權、全球性、有限且不可撤銷的使用權限，僅限於美國政府為傳輸此資料所訂合約所允許之範圍，並基於履行該合約之目的方可使用。除非本文另有規定，否則未經 NetApp Inc. 事前書面許可，不得逕行使用、揭露、重製、修改、履行或展示該資料。美國政府授予國防部之許可權利，僅適用於 DFARS 條款 252.227-7015(b)（2014 年 2 月）所述權利。

商標資訊

NETAPP、NETAPP 標誌及 http://www.netapp.com/TM 所列之標章均為 NetApp, Inc. 的商標。文中所涉及的所有其他公司或產品名稱，均為其各自所有者的商標，不得侵犯。