MLRun Pipeline with Iguazio

NetApp Solutions

NetApp
July 19, 2022

Table of Contents

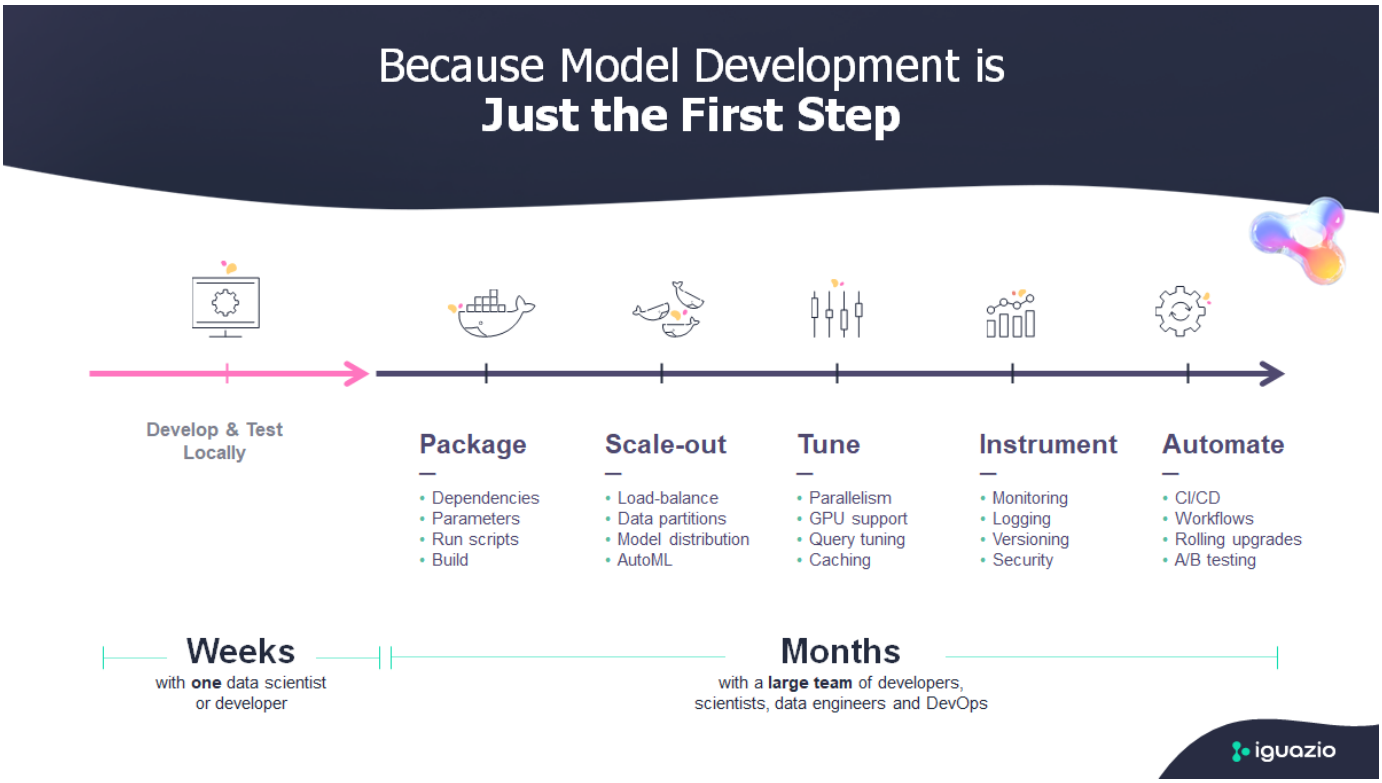
MLRun Pipeline with Iguazio

TR-4834: NetApp and Iguazio for MLRun Pipeline

Rick Huang, David Arnette, NetApp
Marcelo Litovsky, Iguazio

This document covers the details of the MLRun pipeline using NetApp ONTAP AI, NetApp AI Control Plane, NetApp Cloud Volumes software, and the Iguazio Data Science Platform. We used Nuclio serverless function, Kubernetes Persistent Volumes, NetApp Cloud Volumes, NetApp Snapshot copies, Grafana dashboard, and other services on the Iguazio platform to build an end-to-end data pipeline for the simulation of network failure detection. We integrated Iguazio and NetApp technologies to enable fast model deployment, data replication, and production monitoring capabilities on premises as well as in the cloud.

The work of a data scientist should be focused on the training and tuning of machine learning (ML) and artificial intelligence (AI) models. However, according to research by Google, data scientists spend ~80% of their time figuring out how to make their models work with enterprise applications and run at scale, as shown in the following image depicting model development in the AI/ML workflow.



To manage end-to-end AI/ML projects, a wider understanding of enterprise components is needed. Although DevOps have taken over the definition, integration, and deployment these types of components, machine learning operations target a similar flow that includes AI/ML projects. To get an idea of what an end-to-end AI/ML pipeline touches in the enterprise, see the following list of required components:

- Storage
- Networking
- Databases
- File systems

- Containers
- Continuous integration and continuous deployment (CI/CD) pipeline
- Development integrated development environment (IDE)
- Security
- Data access policies
- Hardware
- Cloud
- Virtualization
- Data science toolsets and libraries

In this paper, we demonstrate how the partnership between NetApp and Iguazio drastically simplifies the development of an end-to-end AI/ML pipeline. This simplification accelerates the time to market for all of your AI/ML applications.

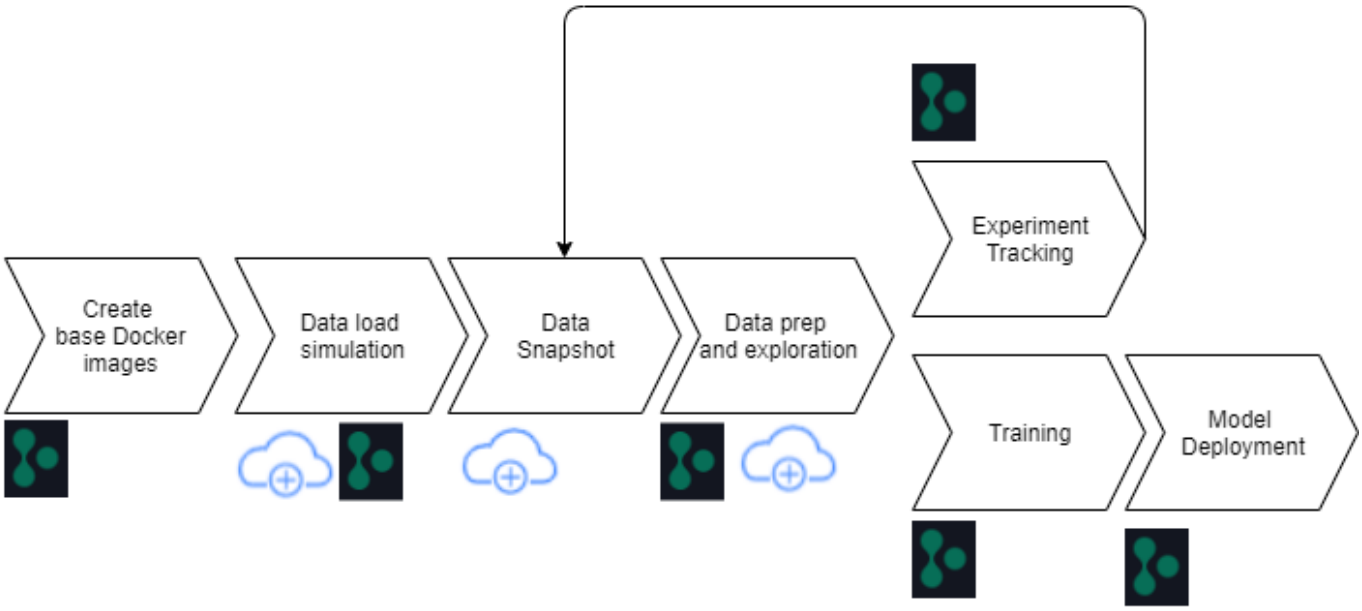
Target Audience

The world of data science touches multiple disciplines in information technology and business.

- The data scientist needs the flexibility to use their tools and libraries of choice.
- The data engineer needs to know how the data flows and where it resides.
- A DevOps engineer needs the tools to integrate new AI/ML applications into their CI/CD pipelines.
- Business users want to have access to AI/ML applications. We describe how NetApp and Iguazio help each of these roles bring value to business with our platforms.

Solution Overview

This solution follows the lifecycle of an AI/ML application. We start with the work of data scientists to define the different steps needed to prep data and train and deploy models. We follow with the work needed to create a full pipeline with the ability to track artifacts, experiment with execution, and deploy to Kubeflow. To complete the full cycle, we integrate the pipeline with NetApp Cloud Volumes to enable data versioning, as seen in the following image.



iguazio

Cloud Volumes ONTAP

Next: Technology Overview

Technology Overview

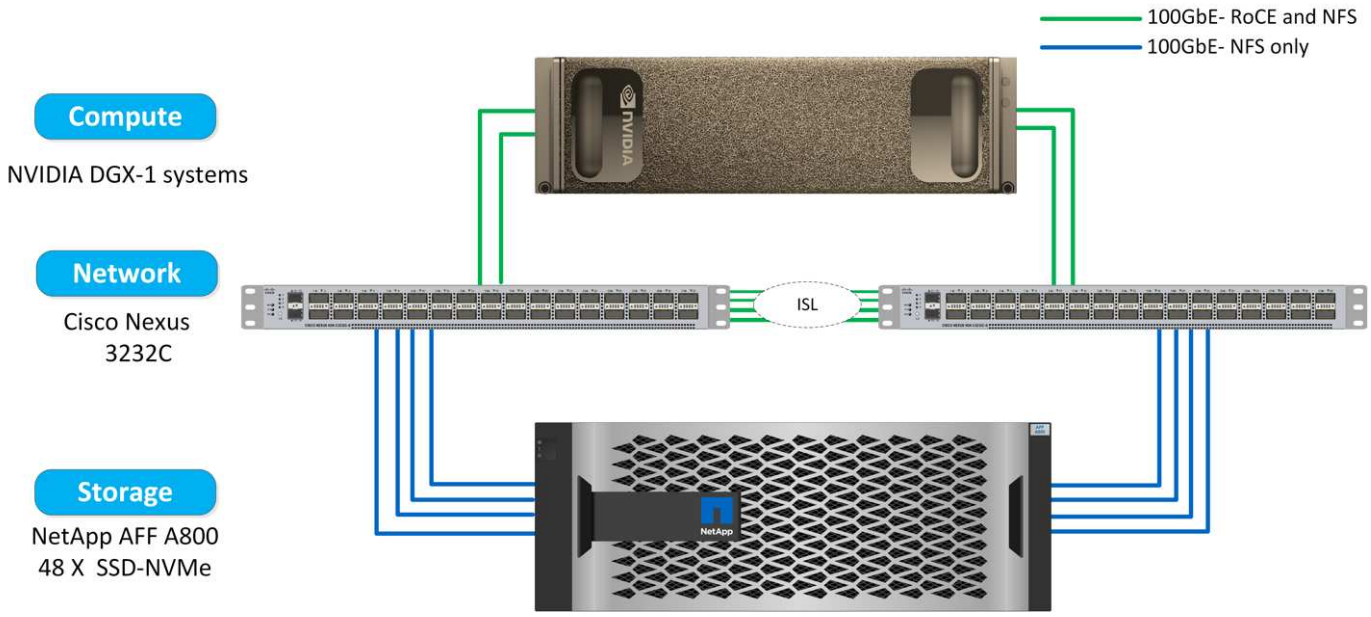
NetApp Overview

NetApp is the data authority for the hybrid cloud. NetApp provides a full range of hybrid cloud data services that simplify management of applications and data across cloud and on-premises environments to accelerate digital transformation. Together with our partners, NetApp empowers global organizations to unleash the full potential of their data to expand customer touch points, foster greater innovation, and optimize their operations.

NetApp ONTAP AI

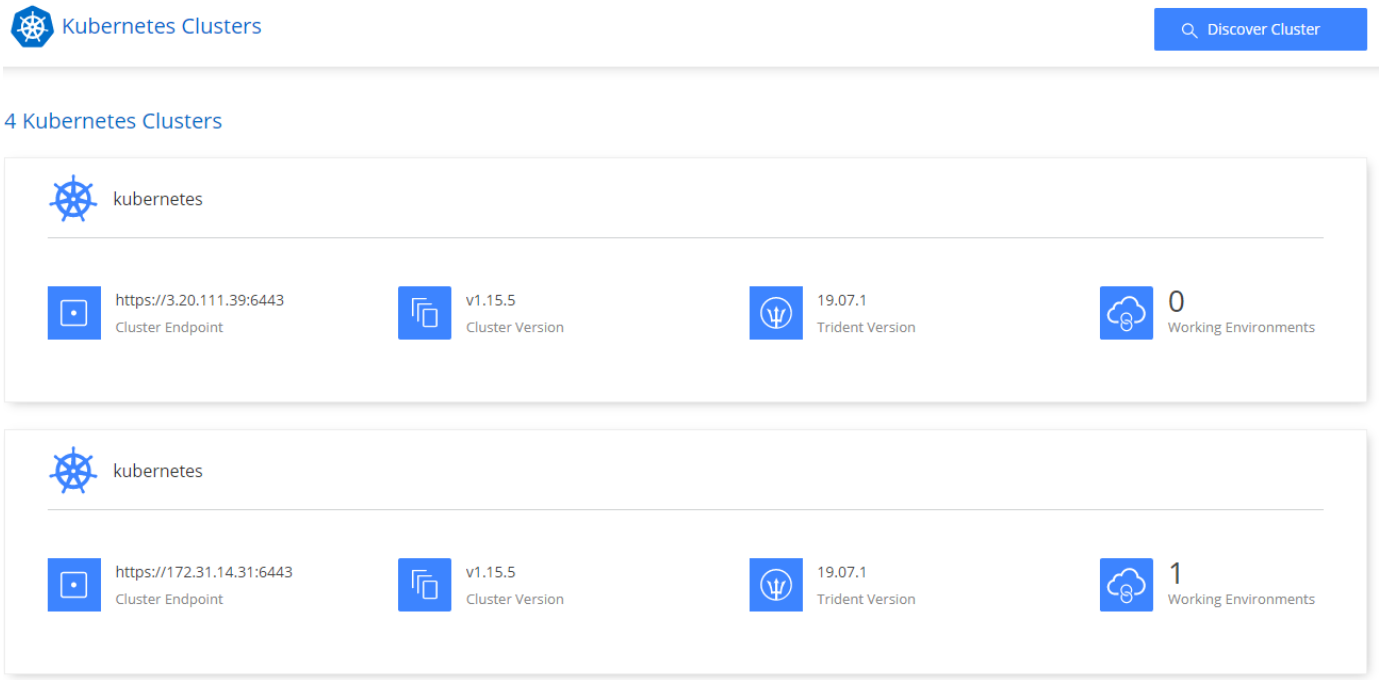
NetApp ONTAP AI, powered by NVIDIA DGX systems and NetApp cloud-connected all-flash storage, streamlines the flow of data reliably and speeds up analytics, training, and inference with your data fabric that spans from edge to core to cloud. It gives IT organizations an architecture that provides the following benefits:

- Eliminates design complexities
- Allows independent scaling of compute and storage
- Enables customers to start small and scale seamlessly
- Offers a range of storage options for various performance and cost pointsNetApp ONTAP AI offers converged infrastructure stacks incorporating NVIDIA DGX-1, a petaflop-scale AI system, and NVIDIA Mellanox high-performance Ethernet switches to unify AI workloads, simplify deployment, and accelerate ROI. We leveraged ONTAP AI with one DGX-1 and NetApp AFF A800 storage system for this technical report. The following image shows the topology of ONTAP AI with the DGX-1 system used in this validation.



NetApp AI Control Plane

The NetApp AI Control Plane enables you to unleash AI and ML with a solution that offers extreme scalability, streamlined deployment, and nonstop data availability. The AI Control Plane solution integrates Kubernetes and Kubeflow with a data fabric enabled by NetApp. Kubernetes, the industry-standard container orchestration platform for cloud-native deployments, enables workload scalability and portability. Kubeflow is an open-source machine-learning platform that simplifies management and deployment, enabling developers to do more data science in less time. A data fabric enabled by NetApp offers uncompromising data availability and portability to make sure that your data is accessible across the pipeline, from edge to core to cloud. This technical report uses the NetApp AI Control Plane in an MLRun pipeline. The following image shows Kubernetes cluster management page where you can have different endpoints for each cluster. We connected NFS Persistent Volumes to the Kubernetes cluster, and the following images show an Persistent Volume connected to the cluster, where NetApp Trident offers persistent storage support and data management capabilities.



Persistent Volumes for Kubernetes

Connected with Kubernetes Cluster

Cloud Volumes ONTAP is connected to 1 Kubernetes cluster. View Cluster

You can connect another Kubernetes cluster to this Cloud Volumes ONTAP system. If the Kubernetes cluster is in a different network than Cloud Volumes ONTAP, specify a custom export policy to provide access to clients.

Kubernetes Cluster

Select Kubernetes Cluster

kubernetes

Custom Export Policy *(Optional)*

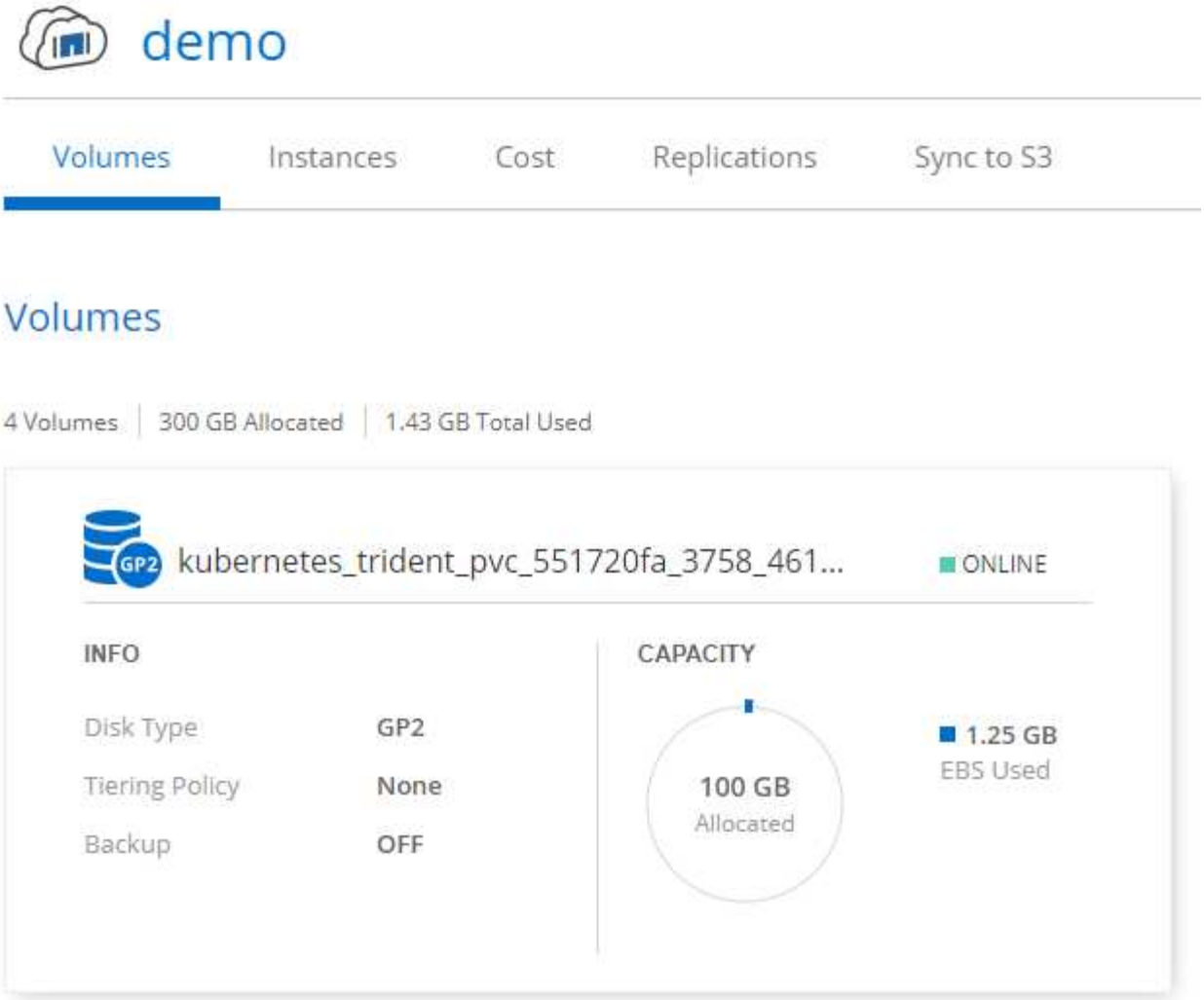
Custom Export Policy

172.31.0.0/16

- [x] Set as default storage class

- (•) NFS
- ( ) iSCSI

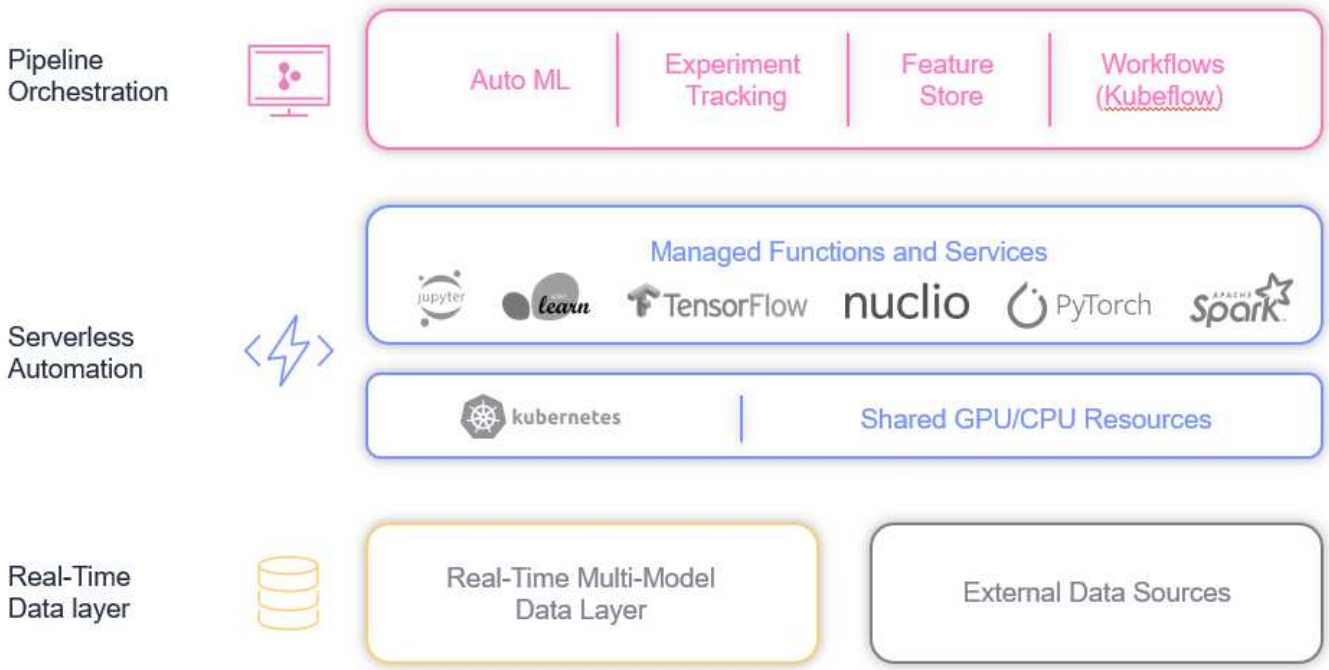
Connect | Cancel



Iguazio Overview

The Iguazio Data Science Platform is a fully integrated and secure data- science platform as a service (PaaS) that simplifies development, accelerates performance, facilitates collaboration, and addresses operational challenges. This platform incorporates the following components, and the Iguazio Data Science Platform is presented in the following image:

- A data-science workbench that includes Jupyter Notebooks, integrated analytics engines, and Python packages
- Model management with experiments tracking and automated pipeline capabilities
- Managed data and ML services over a scalable Kubernetes cluster
- Nuclio, a real-time serverless functions framework
- An extremely fast and secure data layer that supports SQL, NoSQL, time-series databases, files (simple objects), and streaming
- Integration with third-party data sources such as NetApp, Amazon S3, HDFS, SQL databases, and streaming or messaging protocols
- Real-time dashboards based on Grafana



Next: Software and Hardware Requirements

Software and Hardware Requirements

Network Configuration

The following is the network configuration requirement for setting up in the cloud:

- The Iguazio cluster and NetApp Cloud Volumes must be in the same virtual private cloud.
- The cloud manager must have access to port 6443 on the Iguazio app nodes.
- We used Amazon Web Services in this technical report. However, users have the option of deploying the solution in any Cloud provider.For on-premises testing in ONTAP AI with NVIDIA DGX-1, we used the Iguazio hosted DNS service for convenience.

Clients must be able to access dynamically created DNS domains. Customers can use their own DNS if desired.

Hardware Requirements

You can install Iguazio on-premises in your own cluster. We have verified the solution in NetApp ONTAP AI with an NVIDIA DGX-1 system. The following table lists the hardware used to test this solution.

Hardware	Quantity
DGX-1 systems	1
NetApp AFF A800 system	1 high-availability (HA) pair, includes 2 controllers and 48 NVMe SSDs (3.8TB or above)
Cisco Nexus 3232C network switches	2

The following table lists the software components required for on-premise testing:

Software	Version or Other Information
NetApp ONTAP data management software	9.7
Cisco NX-OS switch firmware	7.0(3)I6(1)
NVIDIA DGX OS	4.4 - Ubuntu 18.04 LTS
Docker container platform	19.03.5
Container version	20.01-tf1-py2
Machine learning framework	TensorFlow 1.15.0
Iguazio	Version 2.8+
ESX Server	6.5

This solution was fully tested with Iguazio version 2.5 and NetApp Cloud Volumes ONTAP for AWS. The Iguazio cluster and NetApp software are both running on AWS.

Software	Version or Type
Iguazio	Version 2.8+
App node	M5.4xlarge
Data node	I3.4xlarge

Next: Network Device Failure Prediction Use Case Summary

Network Device Failure Prediction Use Case Summary

This use case is based on an Iguazio customer in the telecommunications space in Asia. With 100K enterprise customers and 125k network outage events per year, there was a critical need to predict and take proactive action to prevent network failures from affecting customers. This solution provided them with the following benefits:

- Predictive analytics for network failures
- Integration with a ticketing system
- Taking proactive action to prevent network failuresAs a result of this implementation of Iguazio, 60% of failures were proactively prevented.

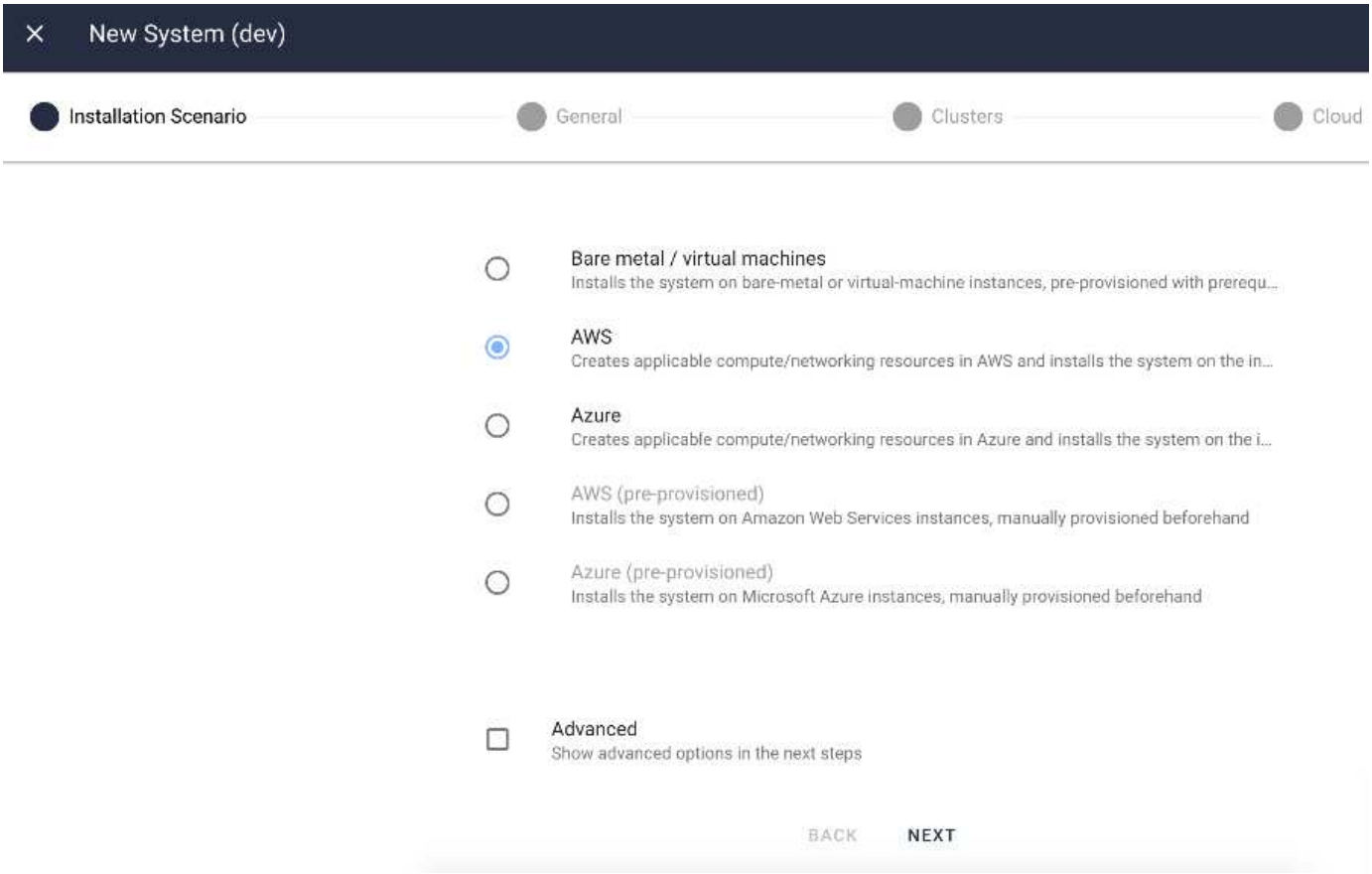
Next: Setup Overview

Setup Overview

Iguazio Installation

Iguazio can be installed on-premises or on a cloud provider. Provisioning can be done as a service and managed by Iguazio or by the customer. In both cases, Iguazio provides a deployment application (Provazio) to deploy and manage clusters.

For on-premises installation, please refer to NVA-1121 for compute, network, and storage setup. On-premises deployment of Iguazio is provided by Iguazio without additional cost to the customer. See this page for DNS and SMTP server configurations. The Provazio installation page is shown as follows.



Next: Configuring Kubernetes Cluster

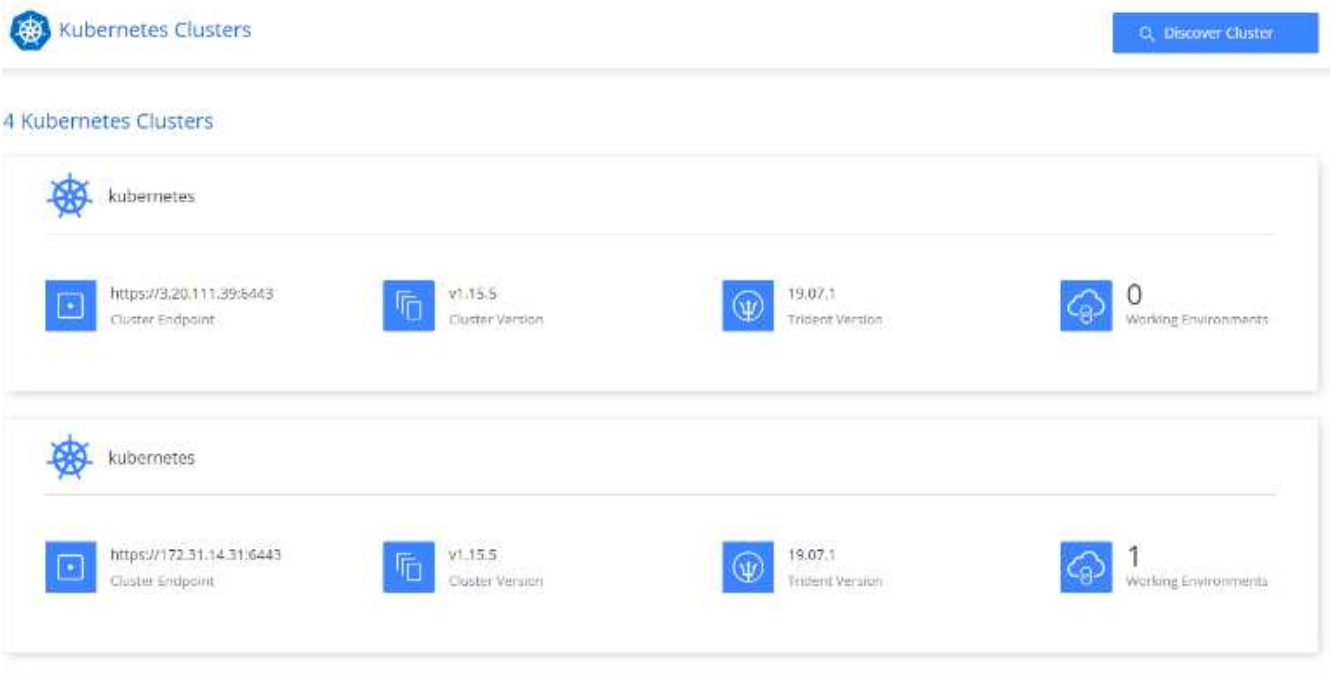
Configuring Kubernetes Cluster

This section is divided into two parts for cloud and on-premises deployment respectively.

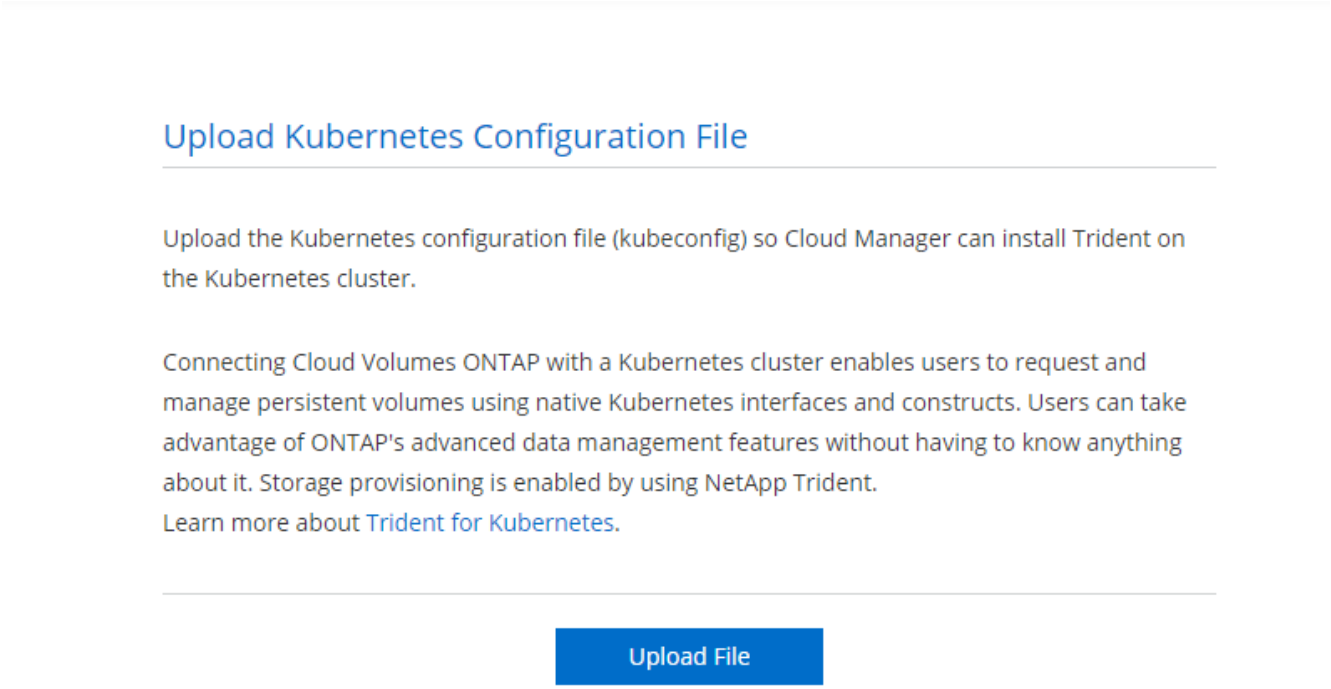
Cloud Deployment Kubernetes Configuration

Through NetApp Cloud Manager, you can define the connection to the Iguazio Kubernetes cluster. Trident requires access to multiple resources in the cluster to make the volume available.

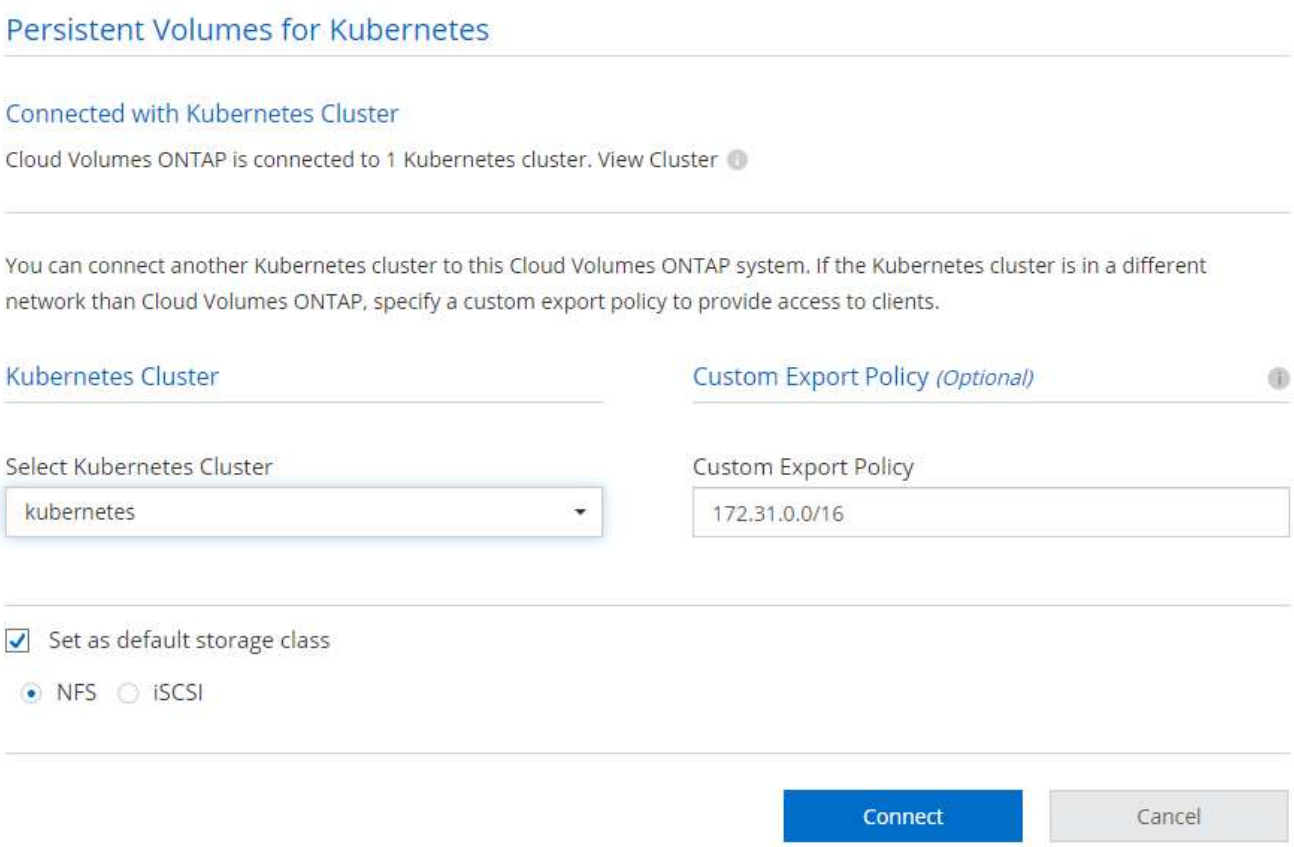
1. To enable access, obtain the Kubernetes config file from one the Iguazio nodes. The file is located under `/home/Iguazio/.kube/config`. Download this file to your desktop.
2. Go to Discover Cluster to configure.



3. Upload the Kubernetes config file. See the following image.



4. Deploy Trident and associate a volume with the cluster. See the following image on defining and assigning a Persistent Volume to the Iguazio cluster.This process creates a Persistent Volume (PV) in Iguazio's Kubernetes cluster. Before you can use it, you must define a Persistent Volume Claim (PVC).



On-Premises Deployment Kubernetes Configuration

For on-premises installation of NetApp Trident, see TR-4798 for details. After configuring your Kubernetes cluster and installing NetApp Trident, you can connect Trident to the Iguazio cluster to enable NetApp data management capabilities, such as taking Snapshot copies of your data and model.

Next: Define Persistent Volume Claim

Define Persistent Volume Claim

1. Save the following YAML to a file to create a PVC of type Basic.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: netapp-file
```

2. Apply the YAML file to your Iguazio Kubernetes cluster.

```
Kubectl -n default-tenant apply -f <your yaml file>
```

Attach NetApp Volume to the Jupyter Notebook

Iguazio offers several managed services to provide data scientists with a full end-to-end stack for development and deployment of AI/ML applications. You can read more about these components at the Iguazio Overview of Application Services and Tools.

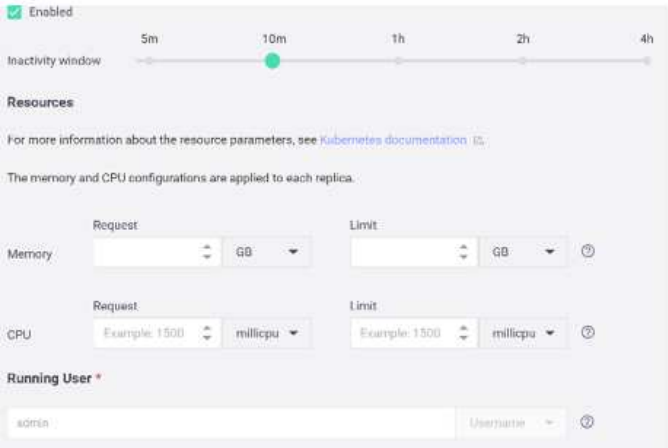
One of the managed services is Jupyter Notebook. Each developer gets its own deployment of a notebook container with the resources they need for development. To give them access to the NetApp Cloud Volume, you can assign the volume to their container and resource allocation, running user, and environment variable settings for Persistent Volume Claims is presented in the following image.

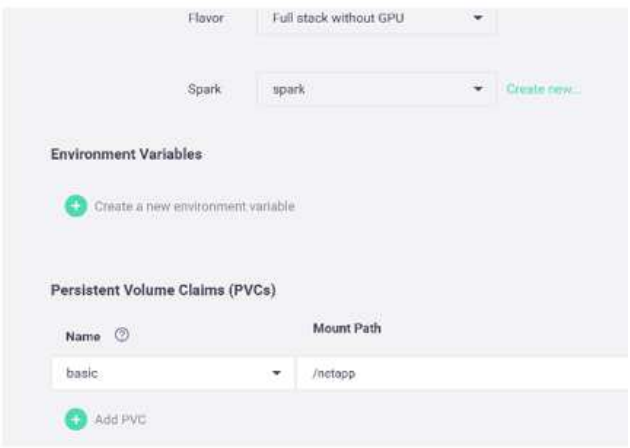
For an on-premises configuration, you can refer to TR-4798 on the Trident setup to enable NetApp ONTAP data management capabilities, such as taking Snapshot copies of your data or model for versioning control. Add the following line in your Trident back- end config file to make Snapshot directories visible:

```
{
    …
    "defaults": {
        "snapshotDir": "true"
    }
}
```

You must create a Trident back- end config file in JSON format, and then run the following Trident command to reference it:

```
tridentctl create backend -f <backend-file>
```





Next: Deploying the Application

Deploying the Application

The following sections describe how to install and deploy the application.

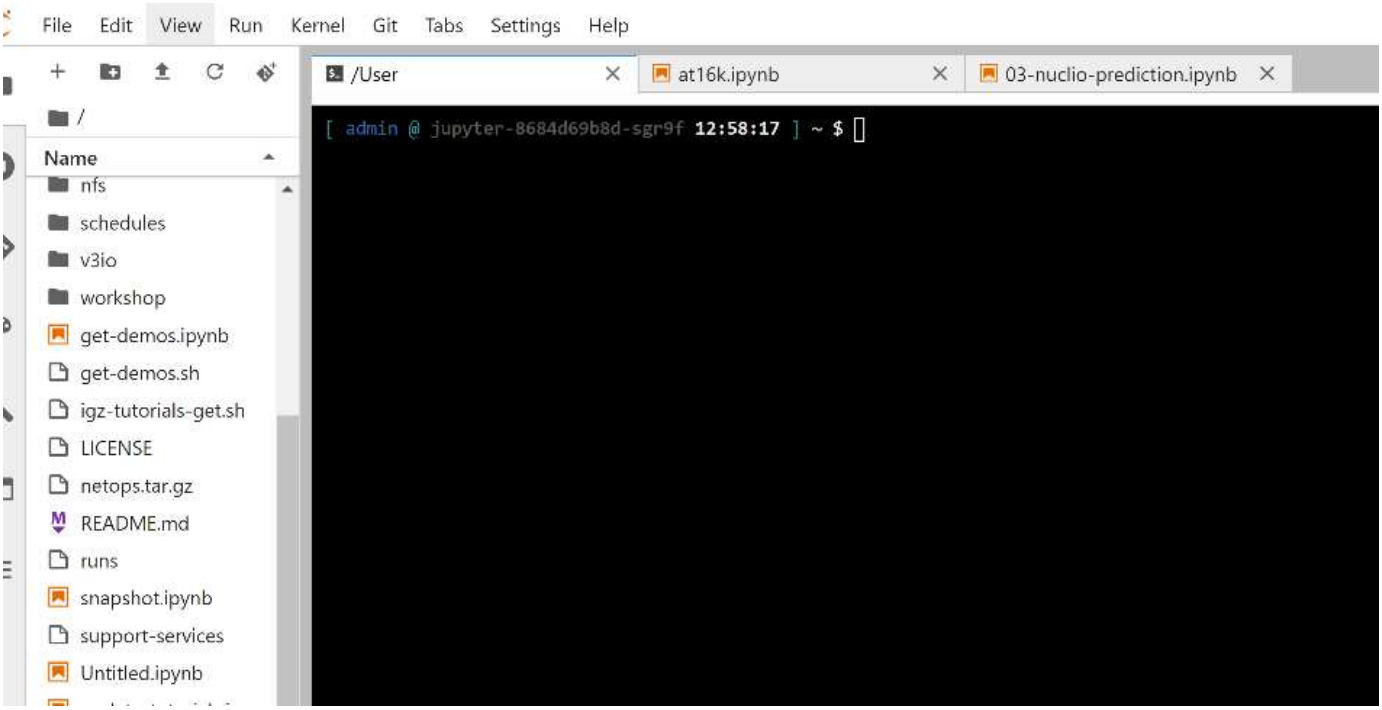
Next: Get Code from GitHub.

Get Code from GitHub

Now that the NetApp Cloud Volume or NetApp Trident volume is available to the Iguazio cluster and the developer environment, you can start reviewing the application.

Users have their own workspace (directory). On every notebook, the path to the user directory is `/User`. The Iguazio platform manages the directory. If you follow the instructions above, the NetApp Cloud volume is available in the `/netapp` directory.

Get the code from GitHub using a Jupyter terminal.



At the Jupyter terminal prompt, clone the project.

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

You should now see the `netops- netapp` folder on the file tree in Jupyter workspace.

Next: Configure Working Environment

Configure Working Environment

Copy the `Notebook set_env-Example.ipynb` as `set_env.ipynb`. Open and edit

`set_env.ipynb`. This notebook sets variables for credentials, file locations, and execution drivers.

If you follow the instructions above, the following steps are the only changes to make:

1. Obtain this value from the Iguazio services dashboard: `docker_registry`

   Example: `docker-registry.default-tenant.app.clusterq.iguaziodev.com:80`

2. Change `admin` to your Iguazio username:

   ```
   IGZ_CONTAINER_PATH = '/users/admin'
   ```

   The following are the ONTAP system connection details. Include the volume name that was generated when Trident was installed. The following setting is for an on-premises ONTAP cluster:

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

   The following setting is for Cloud Volumes ONTAP:

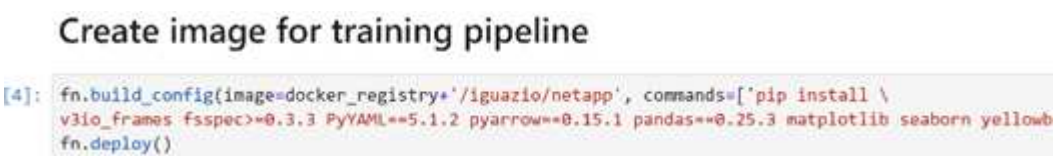
```
MANAGER=ontapClusterMgmtHostname
svm='svm'
email='email'
password=ontapClusterAdminPassword
weid="weid"
volume=sourceVolumeName
```

Create Base Docker Images

Everything you need to build an ML pipeline is included in the Iguazio platform. The developer can define the specifications of the Docker images required to run the pipeline and execute the image creation from Jupyter Notebook. Open the notebook `create- images.ipynb` and Run All Cells.

This notebook creates two images that we use in the pipeline.

- `iguazio/netapp`. Used to handle ML tasks.



- `netapp/pipeline`. Contains utilities to handle NetApp Snapshot copies.

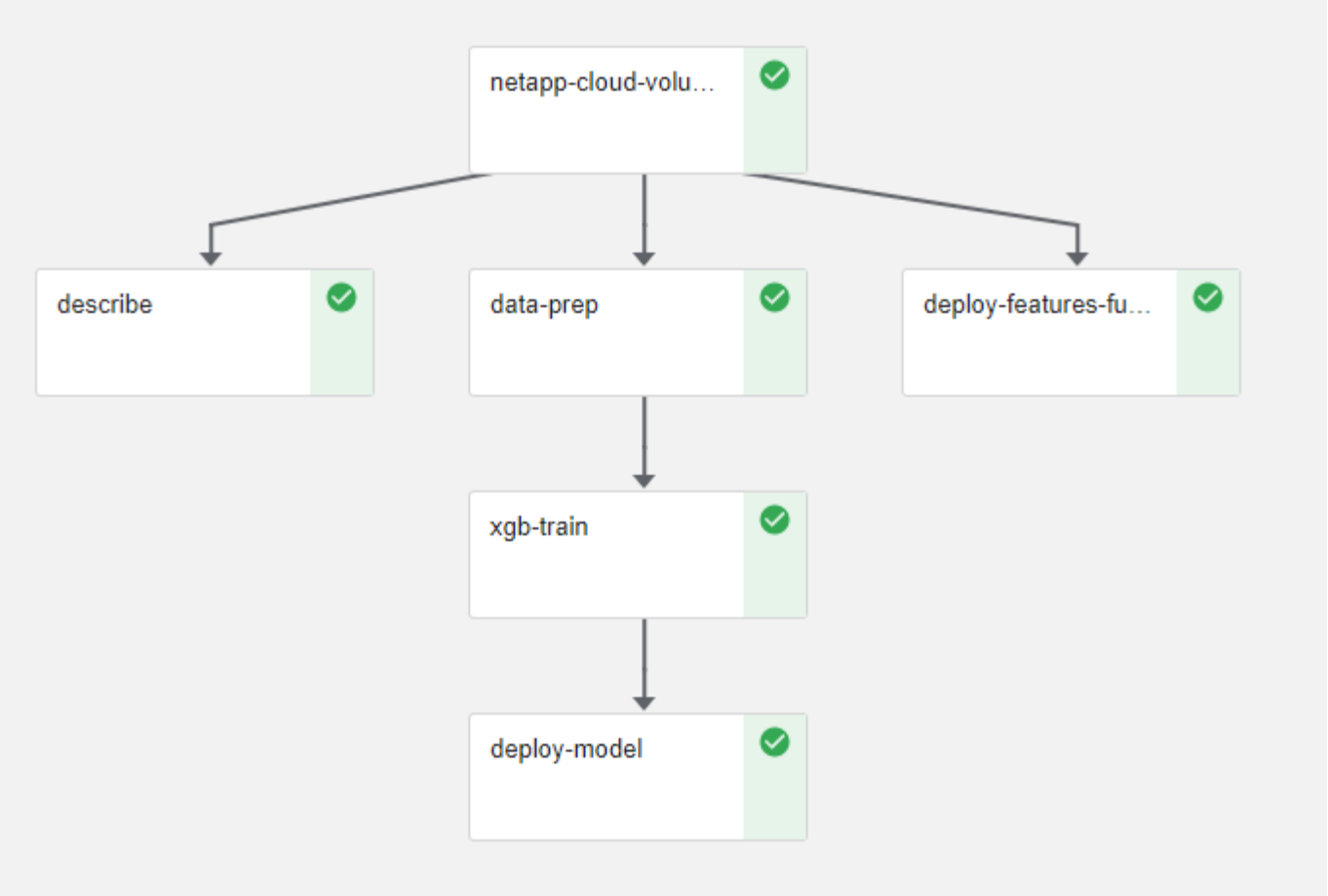
Review Individual Jupyter Notebooks

The following table lists the libraries and frameworks we used to build this task. All these components have been fully integrated with Iguazio's role- based access and security controls.

Libraries/Framework	Description
MLRun	An managed by Iguazio to enable the assembly, execution, and monitoring of an ML/AI pipeline.
Nuclio	A serverless functions framework integrated with Iguazio. Also available as an open-source project managed by Iguazio.
Kubeflow	A Kubernetes-based framework to deploy the pipeline. This is also an open-source project to which Iguazio contributes. It is integrated with Iguazio for added security and integration with the rest of the infrastructure.
Docker	A Docker registry run as a service in the Iguazio platform. You can also change this to connect to your registry.
NetApp Cloud Volumes	Cloud Volumes running on AWS give us access to large amounts of data and the ability to take Snapshot copies to version the datasets used for training.
Trident	Trident is an open-source project managed by NetApp. It facilitates the integration with storage and compute resources in Kubernetes.

We used several notebooks to construct the ML pipeline. Each notebook can be tested individually before being brought together in the pipeline. We cover each notebook individually following the deployment flow of this demonstration application.

The desired result is a pipeline that trains a model based on a Snapshot copy of the data and deploys the model for inference. A block diagram of a completed MLRun pipeline is shown in the following image.



Deploy Data Generation Function

This section describes how we used Nuclio serverless functions to generate network device data. The use case is adapted from an Iguazio client that deployed the pipeline and used Iguazio services to monitor and predict network device failures.

We simulated data coming from network devices. Executing the Jupyter notebook `data- generator.ipynb` creates a serverless function that runs every 10 minutes and generates a Parquet file with new data. To deploy the function, run all the cells in this notebook. See the Nuclio website to review any unfamiliar components in this notebook.

A cell with the following comment is ignored when generating the function. Every cell in the notebook is assumed to be part of the function. Import the Nuclio module to enable `%nuclio magic`.

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

In the spec for the function, we defined the environment in which the function executes, how it is triggered, and the resources it consumes.

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

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

The `init_context` function is invoked by the Nuclio framework upon initialization of the function.

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

Any code not in a function is invoked when the function initializes. When you invoke it, a handler function is executed. You can change the name of the handler and specify it in the function spec.

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

You can test the function from the notebook prior to deployment.

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

The function can be deployed from the notebook or it can be deployed from a CI/CD pipeline (adapting this code).

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

Pipeline Notebooks

These notebooks are not meant to be executed individually for this setup. This is just a review of each notebook. We invoked them as part of the pipeline. To execute them individually, review the MLRun documentation to execute them as Kubernetes jobs.

snap_cv.ipynb

This notebook handles the Cloud Volume Snapshot copies at the beginning of the pipeline. It passes the name of the volume to the pipeline context. This notebook invokes a shell script to handle the Snapshot copy. While running in the pipeline, the execution context contains variables to help locate all files needed for execution.

While writing this code, the developer does not have to worry about the file location in the container that executes it. As described later, this application is deployed with all its dependencies, and it is the definition of the pipeline parameters that provides the execution context.

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

The created Snapshot copy location is placed in the MLRun context to be consumed by steps in the pipeline.

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

The next three notebooks are run in parallel.

data-prep.ipynb

Raw metrics must be turned into features to enable model training. This notebook reads the raw metrics from the Snapshot directory and writes the features for model training to the NetApp volume.

When running in the context of the pipeline, the input `DATA_DIR` contains the Snapshot copy location.

```
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')))
```

describe.ipynb

To visualize the incoming metrics, we deploy a pipeline step that provides plots and graphs that are available through the Kubeflow and MLRun UIs. Each execution has its own version of this visualization tool.

```
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-feature-function.ipynb

We continuously monitor the metrics looking for anomalies. This notebook creates a serverless function that generates the features need to run prediction on incoming metrics. This notebook invokes the creation of the function. The function code is in the notebook `data- prep.ipynb`. Notice that we use the same notebook as a step in the pipeline for this purpose.

training.ipynb

After we create the features, we trigger the model training. The output of this step is the model to be used for inferencing. We also collect statistics to keep track of each execution (experiment).

For example, the following command enters the accuracy score into the context for that experiment. This value

is visible in Kubeflow and MLRun.

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

deploy-inference-function.ipynb

The last step in the pipeline is to deploy the model as a serverless function for continuous inferencing. This notebook invokes the creation of the serverless function defined in `nuclio-inference- function.ipynb`.

Review and Build Pipeline

The combination of running all the notebooks in a pipeline enables the continuous run of experiments to reassess the accuracy of the model against new metrics. First, open the `pipeline.ipynb` notebook. We take you through details that show how NetApp and Iguazio simplify the deployment of this ML pipeline.

We use MLRun to provide context and handle resource allocation to each step of the pipeline. The MLRun API service runs in the Iguazio platform and is the point of interaction with Kubernetes resources. Each developer cannot directly request resources; the API handles the requests and enables access controls.

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

The pipeline can work with NetApp Cloud Volumes and on-premises volumes. We built this demonstration to use Cloud Volumes, but you can see in the code the option to run on-premises.

```
# 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]
```

The first action needed to turn a Jupyter notebook into a Kubeflow step is to turn the code into a function. A function has all the specifications required to run that notebook. As you scroll down the notebook, you can see that we define a function for every step in the pipeline.

Part of the Notebook	**Description**
<code_to_function> (part of the MLRun module)	Name of the function: Project name. used to organize all project artifacts. This is visible in the MLRun UI. Kind. In this case, a Kubernetes job. This could be Dask, mpi, sparkk8s, and more. See the MLRun documentation for more details. File. The name of the notebook. This can also be a location in Git (HTTP).
image	The name of the Docker image we are using for this step. We created this earlier with the create-image.ipynb notebook.
volume_mounts & volumes	Details to mount the NetApp Cloud Volume at run time.

We also define parameters for the steps.

```
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
            }
```

After you have the function definition for all steps, you can construct the pipeline. We use the `kfp` module to make this definition. The difference between using MLRun and building on your own is the simplification and shortening of the coding.

The functions we defined are turned into step components using the `as_step` function of MLRun.

Snapshot Step Definition

Initiate a Snapshot function, output, and mount v3io as source:

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

Parameters	Details
NewTask	NewTask is the definition of the function run.
(MLRun module)	Handler. Name of the Python function to invoke. We used the name handler in the notebook, but it is not required. params. The parameters we passed to the execution. Inside our code, we use context.get_param (‘PARAMETER’) to get the values.

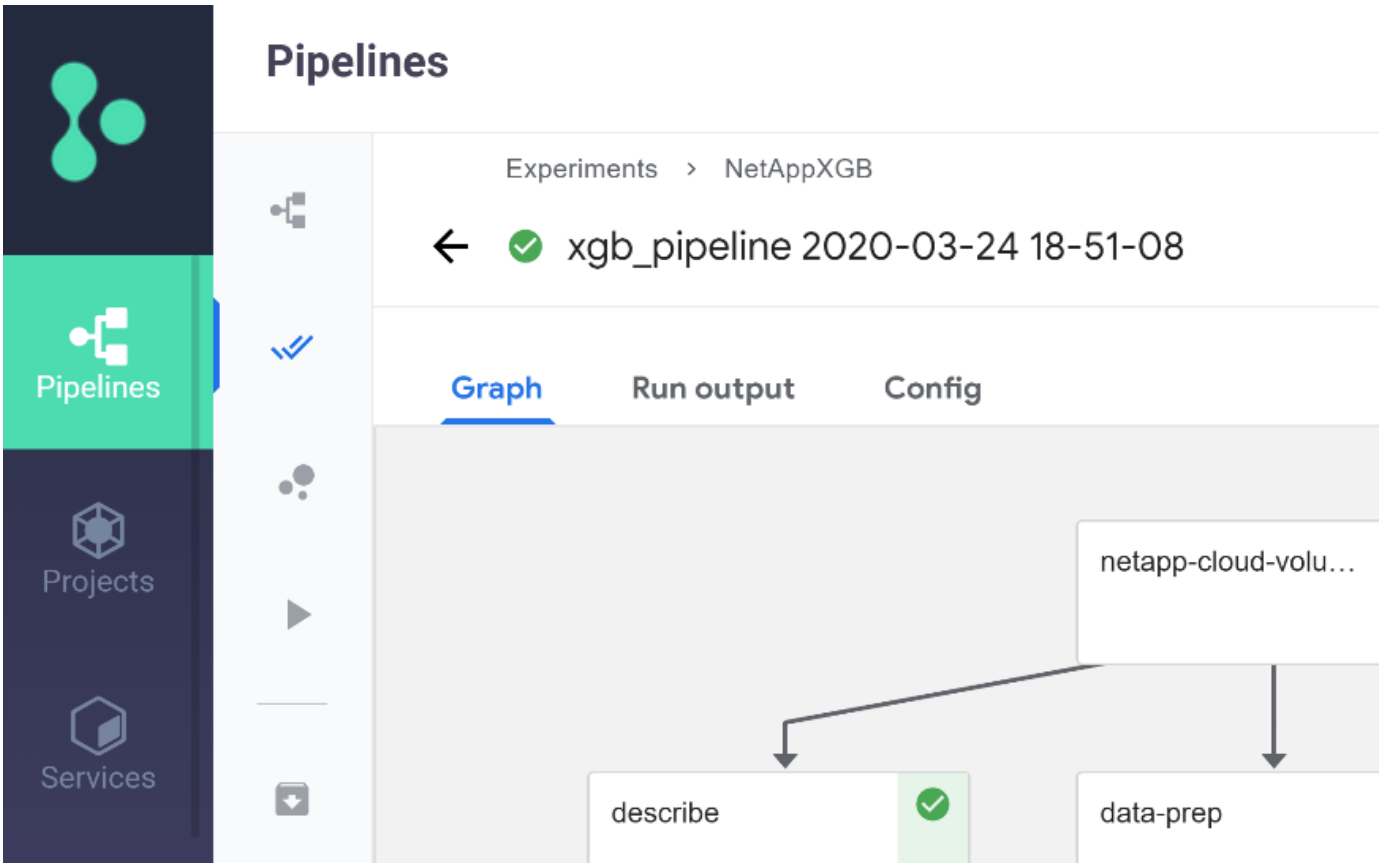
Parameters	Details
as_step	Name. Name of the Kubeflow pipeline step. outputs. These are the values that the step adds to the dictionary on completion. Take a look at the snap_cv.ipynb notebook. mount_v3io(). This configures the step to mount /User for the user executing the pipeline.

```
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)
```

Parameters	Details
inputs	You can pass to a step the outputs of a previous step. In this case, snap.outputs['snapVolumeDetails'] is the name of the Snapshot copy we created on the snap step.
out_path	A location to place artifacts generating using the MLRun module log_artifacts.

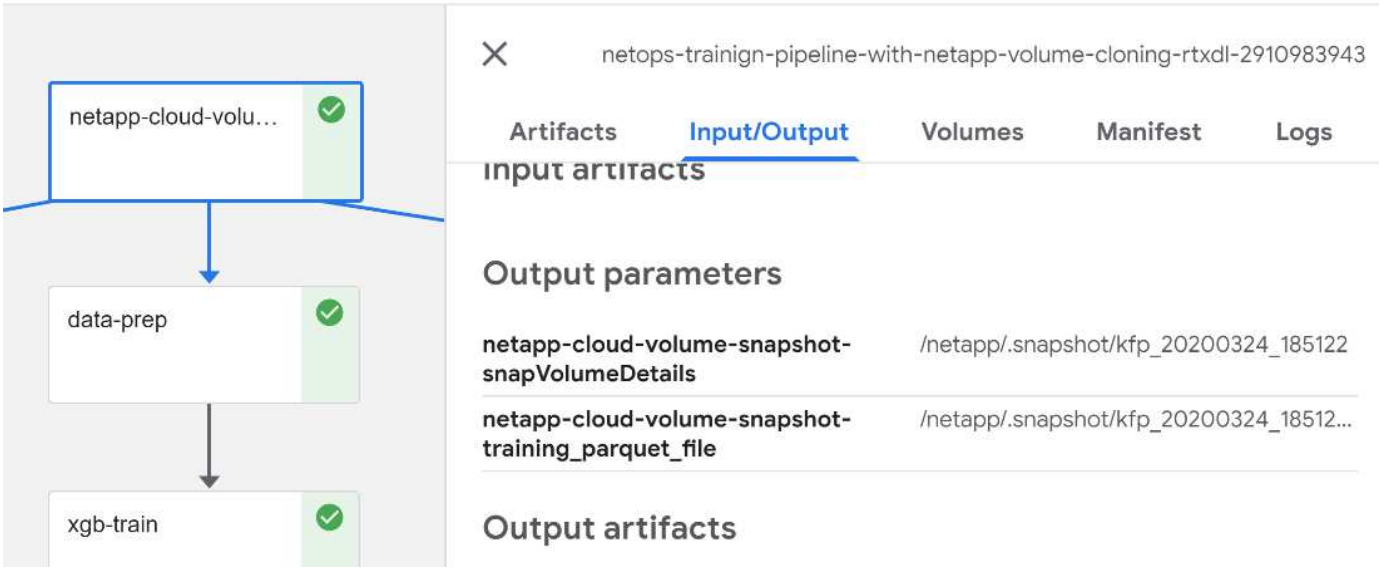
You can run `pipeline.ipynb` from top to bottom. You can then go to the Pipelines tab from the Iguazio dashboard to monitor progress as seen in the Iguazio dashboard Pipelines tab.



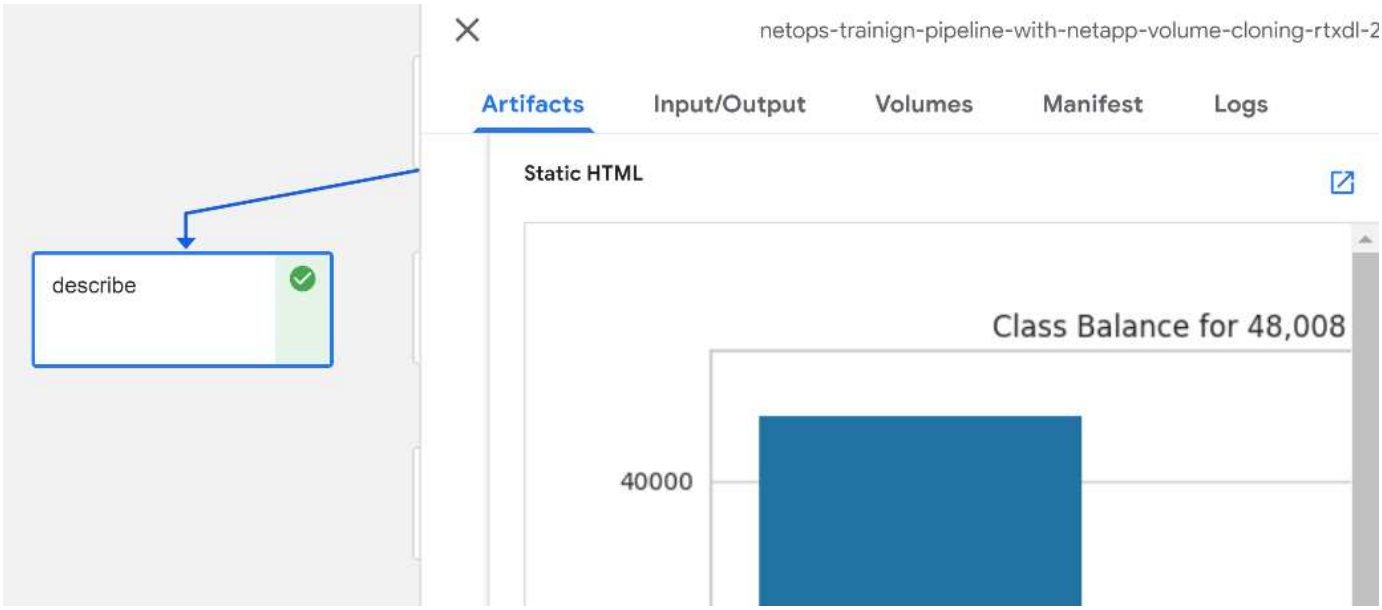
Because we logged the accuracy of training step in every run, we have a record of accuracy for each experiment, as seen in the record of training accuracy.

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

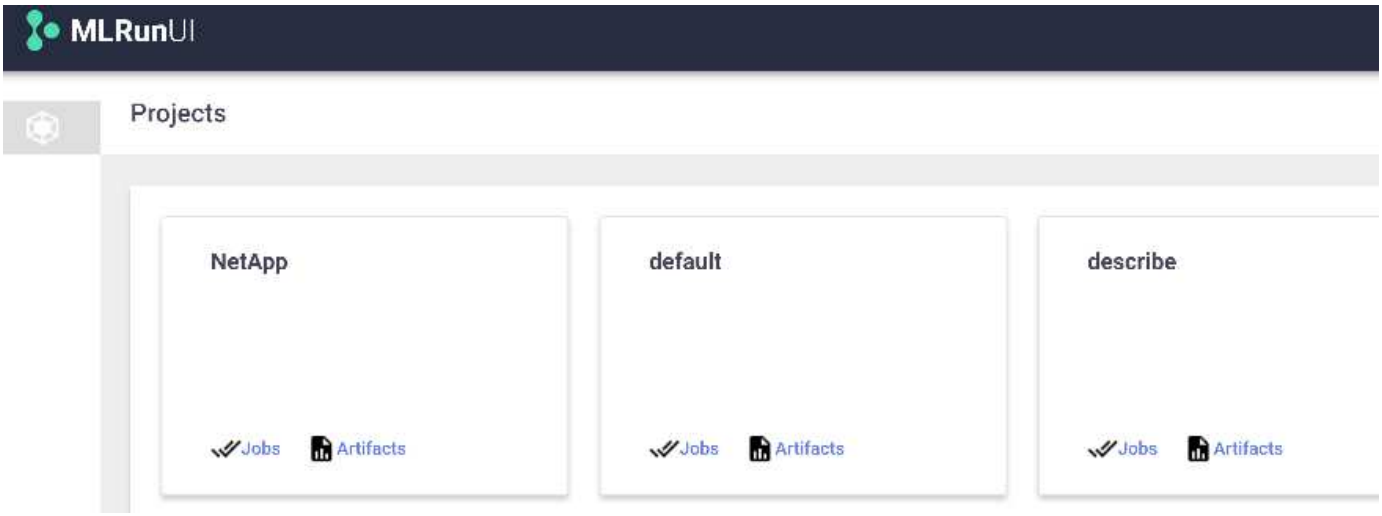
If you select the Snapshot step, you can see the name of the Snapshot copy that was used to run this experiment.



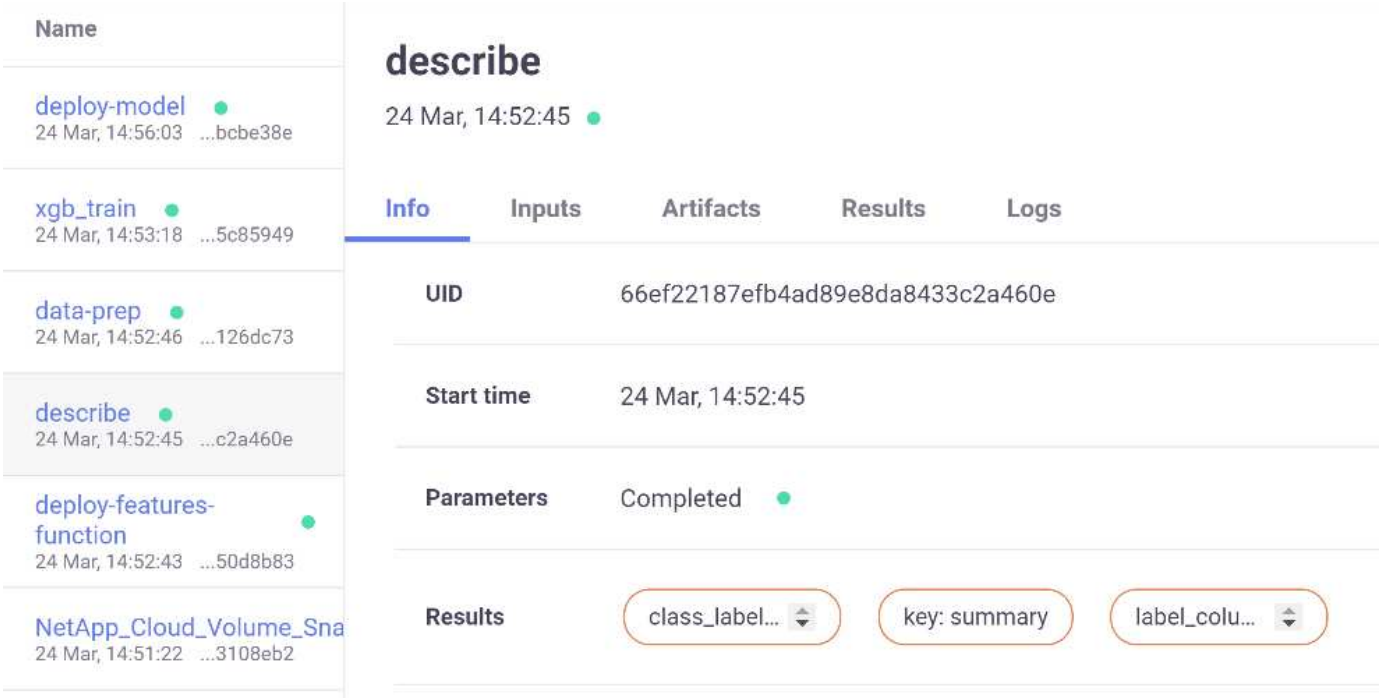
The described step has visual artifacts to explore the metrics we used. You can expand to view the full plot as seen in the following image.



The MLRun API database also tracks inputs, outputs, and artifacts for each run organized by project. An example of inputs, outputs, and artifacts for each run can be seen in the following image.



For each job, we store additional details.



There is more information about MLRun than we can cover in this document. AI artifacts, including the definition of the steps and functions, can be saved to the API database, versioned, and invoked individually or as a full project. Projects can also be saved and pushed to Git for later use. We encourage you to learn more at the MLRun GitHub site.

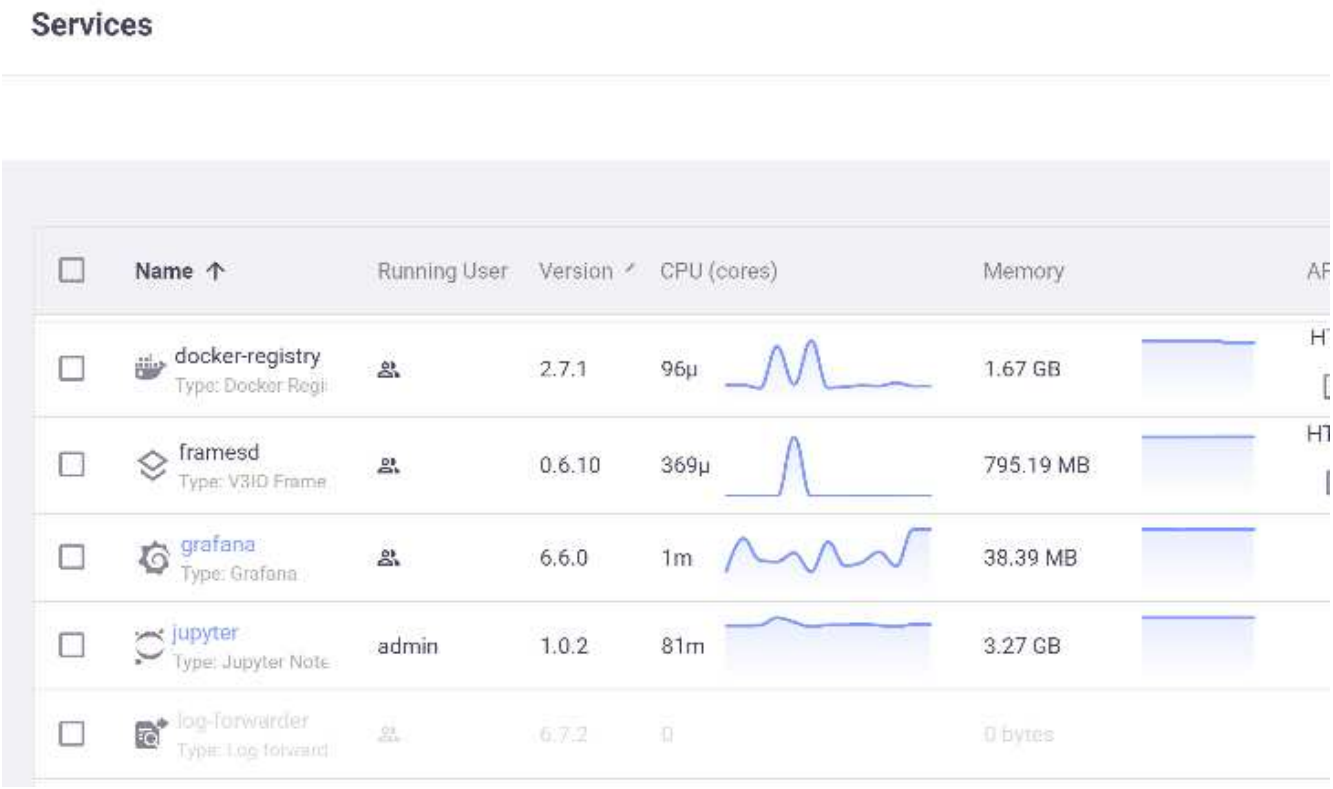
Next: Deploy Grafana Dashboard

Deploy Grafana Dashboard

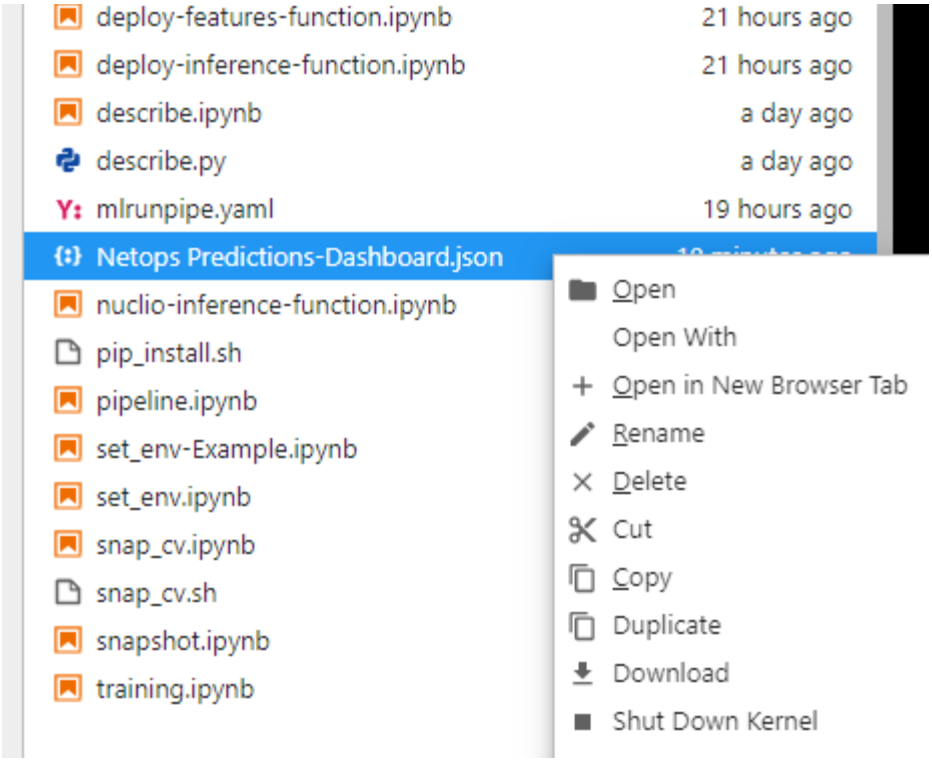
After everything is deployed, we run inferences on new data. The models predict failure on network device equipment. The results of the prediction are stored in an Iguazio TimeSeries table. You can visualize the results with Grafana in the platform integrated with Iguazio's security and data access policy.

You can deploy the dashboard by importing the provided JSON file into the Grafana interfaces in the cluster.

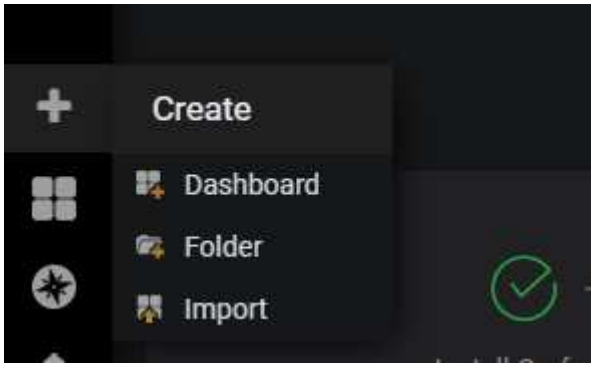
1. To verify that the Grafana service is running, look under Services.



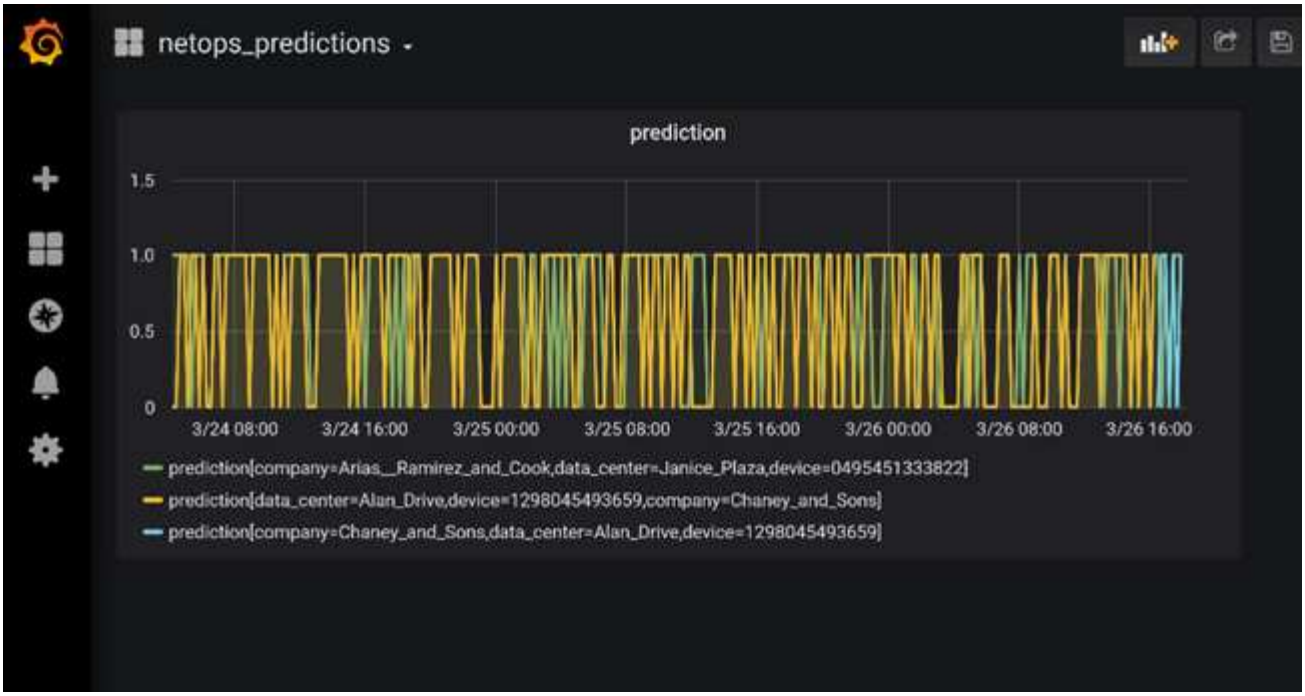
2. If it is not present, deploy an instance from the Services section:
   a. Click New Service.
   b. Select Grafana from the list.
   c. Accept the defaults.
   d. Click Next Step.
   e. Enter your user ID.
   f. Click Save Service.
   g. Click Apply Changes at the top.
3. To deploy the dashboard, download the file `NetopsPredictions-Dashboard.json` through the Jupyter interface.



4. Open Grafana from the Services section and import the dashboard.



5. Click Upload `*.json` File and select the file that you downloaded earlier (`NetopsPredictions-Dashboard.json`). The dashboard displays after the upload is completed.



Deploy Cleanup Function

When you generate a lot of data, it is important to keep things clean and organized. To do so, deploy the cleanup function with the `cleanup.ipynb` notebook.

Benefits

NetApp and Iguazio speed up and simplify the deployment of AI and ML applications by building in essential frameworks, such as Kubeflow, Apache Spark, and TensorFlow, along with orchestration tools like Docker and Kubernetes. By unifying the end-to-end data pipeline, NetApp and Iguazio reduce the latency and complexity inherent in many advanced computing workloads, effectively bridging the gap between development and operations. Data scientists can run queries on large datasets and securely share data and algorithmic models with authorized users during the training phase. After the containerized models are ready for production, you can easily move them from development environments to operational environments.

Next: Conclusion

Conclusion

When building your own AI/ML pipelines, configuring the integration, management, security, and accessibility of the components in an architecture is a challenging task. Giving developers access and control of their environment presents another set of challenges.

The combination of NetApp and Iguazio brings these technologies together as managed services to accelerate technology adoption and improve the time to market for new AI/ML applications.

Next: Where to Find Additional Information