



nodeflux

# A Scalable Training Approach Using Kubeflow

[www.nodeflux.io](http://www.nodeflux.io)

I  
N  
T  
R  
O  
D  
U  
C  
I  
N  
G



nodeflux

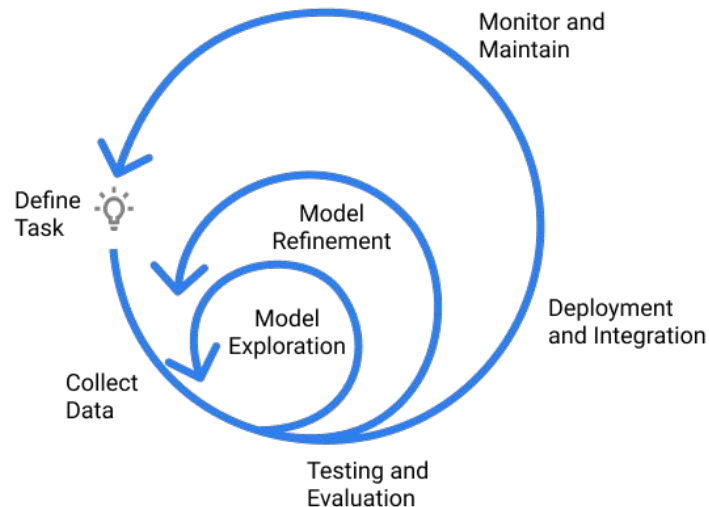
**Alvin Prayuda Juniarta  
Dwiyanoro**

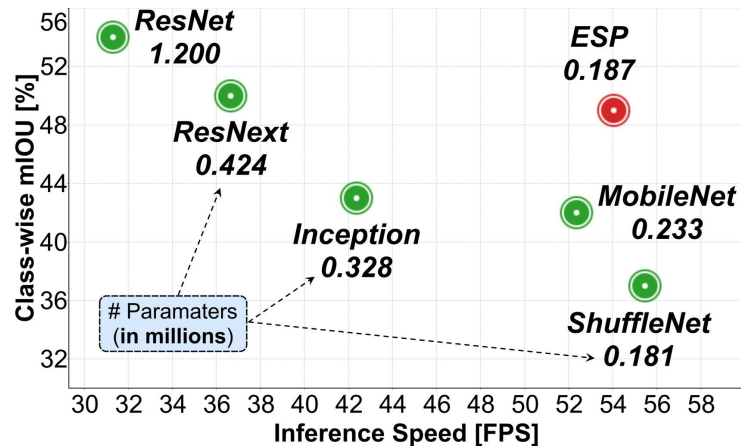
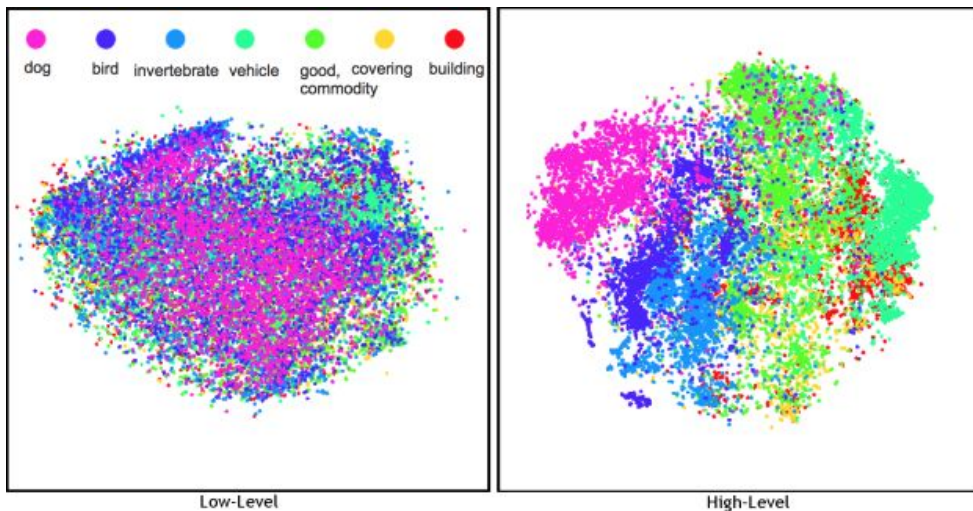
AI Research Group Lead  
of Nodeflux

# Typical Problem

- Iterative training process
- Multiple training resources
- Data migration
- Training process observation

## Machine Learning Development Lifecycle



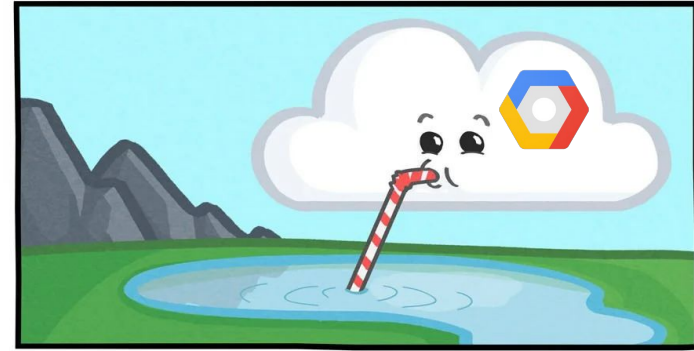


## Iterative Training Process?

- Same problem definition ( classification, regression, object detection ), different hyperparameters
  - Data?
  - Method selection?
  - Number of class?
  - Data augmentation?
  - Feature Engineering?

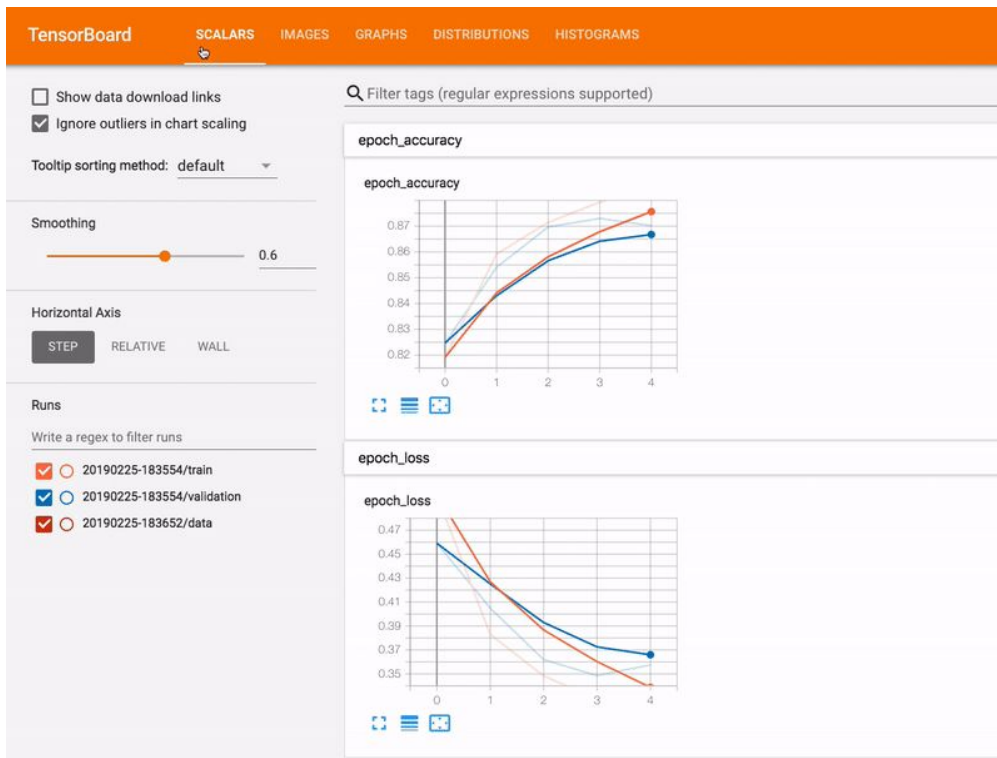
# Data Migration

- Huge size dataset
- Prototype in laptop/pc, train in server/cloud?
  - In our case :
    - local development : Indonesia
    - available training server : US ( for NVIDIA P100/V100)



MS COCO dataset : 25.2 GB





# Training Process Observation

- Loss and metrics visualization
- Error logging



# The Solutions

## The Orchestrator



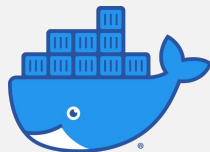
## The Observer



## The Tracker



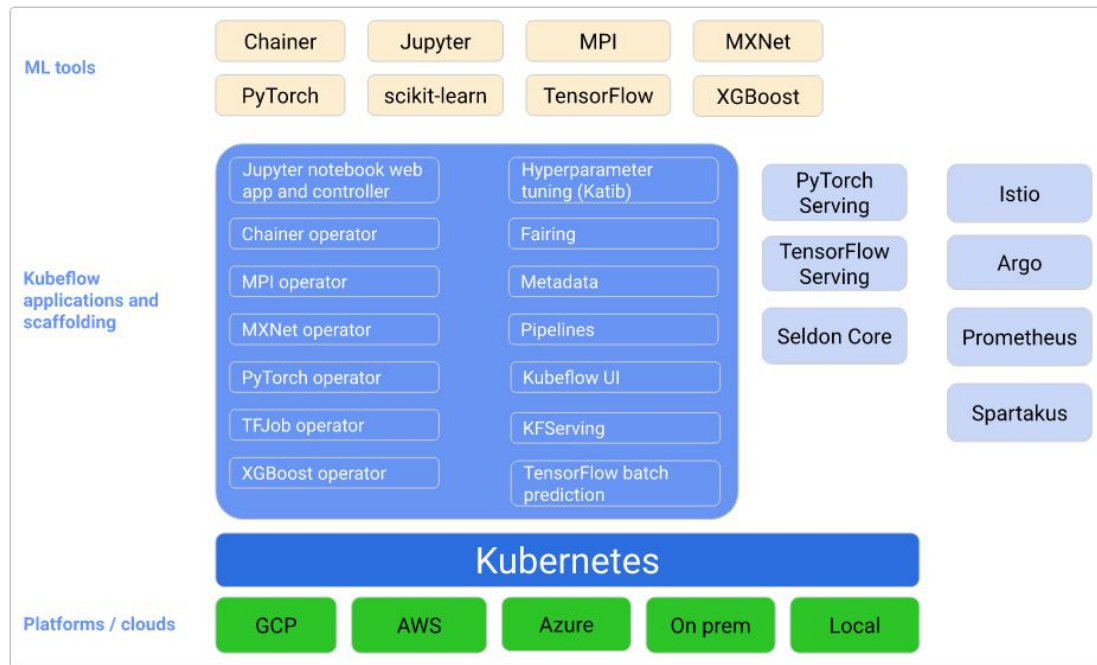
# The Orchestrator





# What is Kubeflow?

- An open source Kubernetes-native platform for developing, orchestrating, deploying, and running scalable and portable ML workloads
- It supports reproducibility and collaboration in ML workflow lifecycles in multiple or hybrid environments ( local → cloud env ) as long as kubernetes exist
- Helps reuse building blocks across different workflows



# Kubeflow UI

nodeflux

← → ↻ ⓘ Not secure | 10.10.10.10:8080/?ns=kubeflow-user

🗺️ Apps 📊 Nodeflux Data 📁 Nodeflux Sour... 📖 Knowledge Re... 💰 Investing



🛠️ kubeflow-user (owner) ▼



Home

Pipelines

Notebook Servers

Katib

Artifact Store

Snapshot Store

Manage Contributors

GitHub

Documentation

Privacy • Usage Reporting  
build version v0.6.2

Dashboard

Activity

## Quick shortcuts

- ⚡ **Upload a pipeline**  
Pipelines
- ⚡ **View all pipeline runs**  
Pipelines
- ⚡ **Create a new Notebook server**  
Notebook Servers
- ⚡ **View Katib Studies**  
Katib
- ⚡ **View Metadata Artifacts**  
Artifact Store

## Recent Notebooks

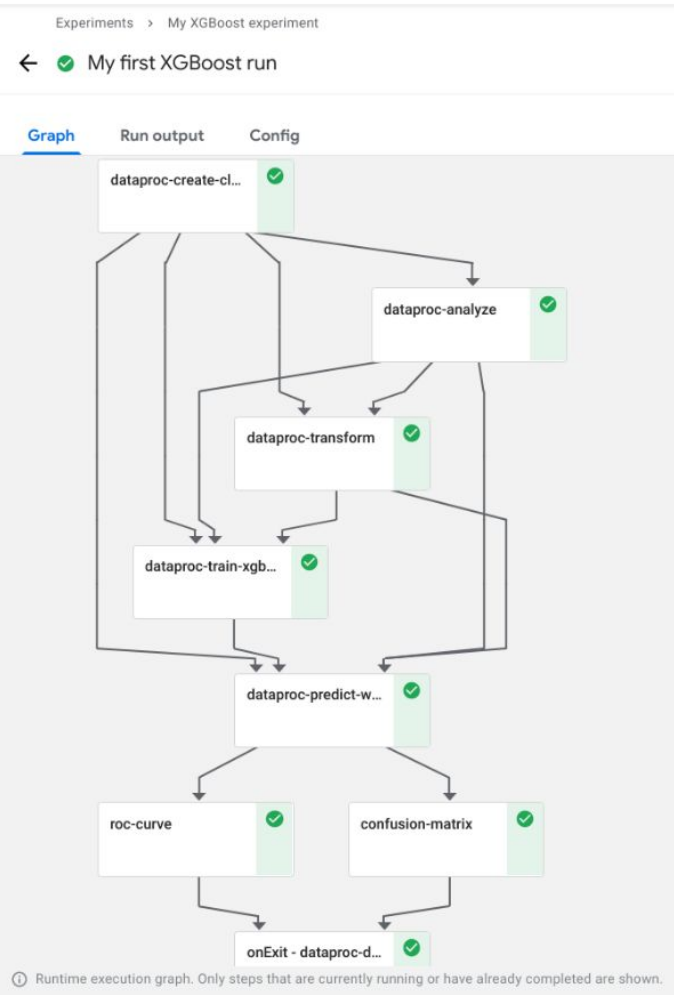
- 📄 **mnist-train-pipeline.tar.gz**  
Accessed 1/29/2020, 12:14:38 PM
- 📄 **Untitled.ipynb**  
Accessed 1/29/2020, 12:12:15 PM
- 📄 **taxi-cab-pipeline-snap.ipynb**  
Accessed 1/29/2020, 12:10:43 PM
- 📄 **mnist-train-snap.py**  
Accessed 1/29/2020, 12:09:50 PM
- 📄 **kubeflow**  
Accessed 1/27/2020, 3:44:11 PM

## Recent Pipelines

- 🔧 **[Sample] Basic - Exit Handler**  
Created 1/15/2020, 6:39:48 PM
- 🔧 **[Sample] Basic - Conditional execution**  
Created 1/15/2020, 6:39:47 PM
- 🔧 **[Sample] Basic - Parallel execution**  
Created 1/15/2020, 6:39:45 PM
- 🔧 **[Sample] Basic - Sequential execution**  
Created 1/15/2020, 6:39:43 PM
- 🔧 **[Sample] ML - TFX - Taxi Tip Prediction Model Trainer**  
Created 1/15/2020, 6:39:40 PM

## Documentation

- Getting Started with Kubeflow**  
Get your machine-learning workflow up and running on Kubeflow
- MiniKF**  
A fast and easy way to deploy Kubeflow locally
- Microk8s for Kubeflow**  
Quickly get Kubeflow running locally on native hypervisors
- Minikube for Kubeflow**  
Quickly get Kubeflow running locally
- Kubeflow on GCP**  
Running Kubeflow on Kubernetes Engine and Google Cloud Platform
- Kubeflow on AWS**  
Running Kubeflow on Elastic Container Service and Amazon Web Services
- Requirements for Kubeflow**  
Get more detailed information about using Kubeflow and its components



# Kubeflow Pipelines

- A platform for building and deploying portable, scalable machine learning (ML) workflows based on Docker containers
  - Provides UI for experiments
  - Engine for scheduling multi-steps ML workflow
  - Interactive Jupyter to interact with pipelines
- The pipelines will consist of several components which is an executable functions inside the docker container

# Local Development

- Using MiniKF to deploy Kubeflow locally ( on laptop )
- MiniKF utilizing vagrant to package the Kubeflow VM
- MiniKF tutorial :
  - <https://medium.com/kubeflow/an-end-to-end-mini-kf-pipeline-on-prem-notebooks-kubeflow-pipeline-on-the-new-minikf-33b7d8e9a836>

```
nodeflux@nodeflux: ~/Documents/nodeflux-repo/res
nodeflux@nodeflux: ~/Documents/nodeflux-repo/resear
→ MiniKF vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Using /home/nodeflux/Documents/nodeflux-repo/research/MiniKF/minikf-user-data.vdi for persistent storage.
==& default: Importing base box 'arrikto/minikf'...
==& default: Generating MAC address for NAT networking...
==& default: Using /home/nodeflux/Documents/nodeflux-repo/research/MiniKF/minikf-user-data.vdi for persistent storage.
==& default: Checking if box 'arrikto/minikf' version '20190918.1.0' is up to date...
==& default: Setting the name of the VM: MiniKF_default_1580275242904_34647
==& default: Clearing any previously set network interfaces...
==& default: Preparing network interfaces based on configuration...
default: Adapter 1: nat
default: Adapter 2: hostonly
==& default: Forwarding ports...
default: 32123 (guest) => 6174 (host) (adapter 1)
default: 22 (guest) => 2222 (host) (adapter 1)
==& default: Using /home/nodeflux/Documents/nodeflux-repo/research/MiniKF/minikf-user-data.vdi for persistent storage.
==& default: Running 'pre-boot' VM customizations...
==& default: Using /home/nodeflux/Documents/nodeflux-repo/research/MiniKF/minikf-user-data.vdi for persistent storage.
==& default: ** Persistent Storage Volume exists, not creating **
==& default: Using /home/nodeflux/Documents/nodeflux-repo/research/MiniKF/minikf-user-data.vdi for persistent storage.
==& default: ** Attaching persistent storage **
==& default: Booting VM...
==& default: Waiting for machine to boot. This may take a few minutes...
```

```
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
default: The guest additions on this VM do not match the installed version of
default: VirtualBox! In most cases this is fine, but in rare cases it can
default: prevent things such as shared folders from working properly. If you see
default: shared folder errors, please make sure the guest additions within the
default: virtual machine match the version of VirtualBox you have installed on
default: your host and reload your VM.
default:
default: Guest Additions Version: 6.0.12_Ubuntu r132055
default: VirtualBox Version: 5.2
==& default: Using /home/nodeflux/Documents/nodeflux-repo/research/MiniKF/minikf-user-data.vdi for persistent storage.
==& default: ** Managing persistent storage **
==& default: Setting hostname...
==& default: Configuring and enabling network interfaces...
==& default: Mounting shared folders...
default: /vagrant => /home/nodeflux/Documents/nodeflux-repo/research/MiniKF

==> default: Machine 'default' has a post 'vagrant up' message. This is a message
==> default: from the creator of the Vagrantfile, and not from Vagrant itself:
==> default:
==> default:     Welcome to MiniKF!
==> default:     Visit http://10.10.10.10/ to get started.
```

# Local Development

- Accessing <http://10.10.10.10> for MiniKF landing page
- We can access Kubeflow and Rok UI page from here
- Rok is used to snapshot the Jupyter volume in which will be used to store the data
- In our case, we only use MiniKF to provide the Kubeflow environment

**Welcome to MiniKF!**  
Use the terminal to start MiniKF

```
*** MiniKF provisioning tool terminated. Press Enter to restart, Ctrl-C to exit...  
█
```

 Credentials

username **user**  
password **12341234**

 Kubeflow

**Connect**

 Rok

**Connect**

Up 

Up 

# **Step 1 : Develop the Docker Image**

# Training Script Development

- Keep in mind, Kubeflow have several conditions
  - Accept argument parse as input
  - Each outputs must be saved as string and stored in local file inside the container

```
app.py
home > nodeflux > Documents > nodeflux-repo > research > MiniKF > mnist_training > app.py > ...
1  from comet_ml import Experiment
2  import argparse
3  from datetime import datetime
4  import tensorflow as tf
5  import os
6
7  os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = 'AI-training-a9aad66abc4e.json'
8
9  experiment = Experiment(api_key="0kZ5vwvCG7xBToH0eptFxyxu",
10 | | | | | | | | | | project_name="test-kubeflow", workspace="hyperion-rg")
11
12  parser = argparse.ArgumentParser()
13  parser.add_argument(
14 | | '--model_file', type=str, required=True, help='Name of the model file.')
15  parser.add_argument(
16 | | '--bucket', type=str, required=True, help='GCS bucket name.')
17  args = parser.parse_args()
18
19  bucket=args.bucket
20  model_file=args.model_file
21
22  model = tf.keras.models.Sequential([
23 |   tf.keras.layers.Flatten(input_shape=(28, 28)),
24 |   tf.keras.layers.Dense(512, activation=tf.nn.relu),
25 |   tf.keras.layers.Dropout(0.2),
26 |   tf.keras.layers.Dense(10, activation=tf.nn.softmax)
27  ])
28
29  model.compile(optimizer='adam',
30 | | | | | | loss='sparse_categorical_crossentropy',
31 | | | | | | metrics=['accuracy'])
32
33  print(model.summary())
```



## Training Script Development

- Keep in mind, Kubeflow have several conditions
  - Accept argument parse as input
  - Each outputs must be saved as string and stored in local file inside the container

```
50
51 model.fit(x_train, y_train, batch_size=32, epochs=5, callbacks=callbacks,
52         validation_data=(x_test, y_test))
53
54
55 model.save(model_file)
56
57 from tensorflow import gfile
58
59 gcs_path = bucket + "/" + model_file
60
61 if gfile.Exists(gcs_path):
62     gfile.Remove(gcs_path)
63
64 gfile.Copy(model_file, gcs_path)
65 with open('/output.txt', 'w') as f:
66     f.write(gcs_path)
67
```

# Build Docker and Push to Registry

- Create Dockerfile
- Build locally and commit to your selected registry ( make sure your Kubeflow can access it later )

Name

Dockerfile  
build\_image.sh  
app.py

Dockerfile X

```
home > nodeflux > Documents > nodeflux-repo > research > M
1 FROM tensorflow/tensorflow:1.15.0-py3
2 RUN pip install comet_ml
3 WORKDIR /app
4 COPY . /app
```

build\_image.sh X

```
home > nodeflux > Documents > nodeflux-repo > research > MiniKF > mnist_training > build_image.sh
1 docker build -t "mnist_training_kf_pipeline" .
2 docker tag "mnist_training_kf_pipeline" "gcr.io/ai-training-250314/mnist_training_kf_pipeline:latest"
3 docker push "gcr.io/ai-training-250314/mnist_training_kf_pipeline:latest"
4 docker image rm "mnist_training_kf_pipeline"
5 docker image rm "gcr.io/ai-training-250314/mnist_training_kf_pipeline:latest"
```

## **Step 2 : Create Reusable Kubeflow Components as a Package**

# Kubeflow Components Development

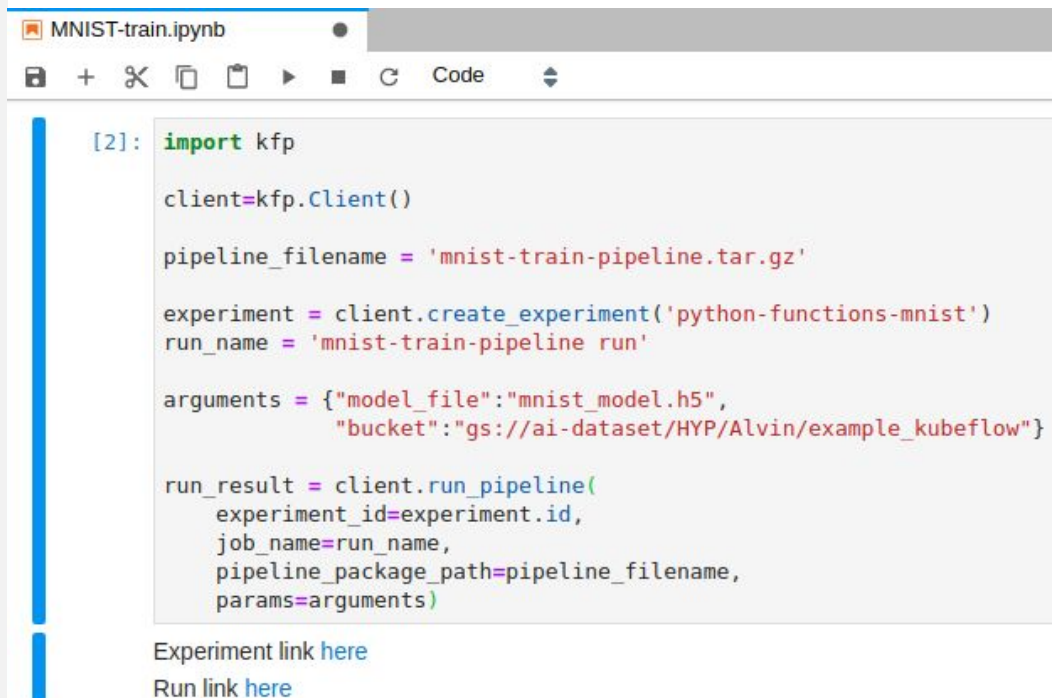
- Kubeflow wrap script interaction inside a container as a python function, we call it ContainerOp
- We can compile the operations and pipeline as a single deployment file ( usually ends with .tar.gz ) and freely distributed to others so that they can use it

```
mnist-train-snap.py x
1 import kfp.dsl as dsl
2 import kfp.gcp as gcp
3
4 def mnist_train_op(model_file, bucket):
5     return dsl.ContainerOp(
6         name="mnist_training_container",
7         image='gcr.io/ai-training-250314/mnist_training_kf_pipeline:latest',
8         command=['python', '/app/app.py'],
9         file_outputs={'outputs': '/output.txt'},
10        arguments=['--bucket', bucket, '--model_file', model_file]
11    )
12
13 # Define the pipeline
14 @dsl.pipeline(
15     name='Mnist pipeline',
16     description='A toy pipeline that performs mnist model training.'
17 )
18 def mnist_container_pipeline(
19     model_file: str = 'mnist_model.h5',
20     bucket: str = 'gs://ai-dataset/HYP/Alvin/example_kubeflow'
21 ):
22     # mnist_train_op(model_file=model_file, bucket=bucket).apply(gcp.use_gcp_secret('user-gcp-sa'))
23     mnist_train_op(model_file=model_file, bucket=bucket)
24
25 if __name__ == '__main__':
26     import kfp.compiler as compiler
27
28     compiler.Compiler().compile(mnist_container_pipeline, 'mnist-train-pipeline.tar.gz')
```

## **Step 3 : Execute the Deployment Package in the Kubeflow Cluster**

# Use the Pre-compiled Component

- Via scripting, possible via UI
- Simple execution, fast orchestration



```
[2]: import kfp

client=kfp.Client()

pipeline_filename = 'mnist-train-pipeline.tar.gz'

experiment = client.create_experiment('python-functions-mnist')
run_name = 'mnist-train-pipeline run'

arguments = {"model_file": "mnist_model.h5",
             "bucket": "gs://ai-dataset/HYP/Alvin/example_kubeflow"}


run_result = client.run_pipeline(
    experiment_id=experiment.id,
    job_name=run_name,
    pipeline_package_path=pipeline_filename,
    params=arguments)
```


Experiment link [here](#)  
Run link [here](#)


# Inspect the Experiments Orchestration

nodeflux

- The executed workflow will appear on the experiments tab
- Running experiment will show the status, duration, etc..

 Pipelines

 Experiments

 Archive

<

Experiments


python-functions-mnist

Refresh

Recurring run configs

0 active

Manage

Experiment description 

Runs

+ Create run


+ Create recurring run

Compare runs

Clone run

Archive

Filter runs

<input type="checkbox"/> Run name	Status	Duration	Pipeline	Recurring Run	Start time ↓
<input type="checkbox"/> mnist-train-pipeline run		0:01:26	[View pipeline]	-	1/29/2020, 6:30:14 PM

Rows per page: 10 < >



# Inspect the Experiments Orchestration

nodeflux

- We can inspect the details of each run
  - Input parameters
  - Output value
  - Logs

The screenshot displays the Nodeflux Experiments interface. On the left, a sidebar contains navigation links: Pipelines, Experiments (highlighted with a checkmark), and Archive. The main content area shows the breadcrumb 'Experiments > python-functions-mnist' and a back arrow followed by 'mnist-train-pipeline run' with a green status icon. Below this are three tabs: Graph, Run output (selected), and Config. The Run output tab shows a single task 'mnist-training-cont...' with a green status icon. On the right, a modal window titled 'mnist-pipeline-r8cwf-2917520262' is open, showing tabs for Artifacts, Input/Output (selected), Volumes, Manifest, and Logs. The Input/Output tab is divided into 'Input parameters' and 'Output parameters' sections.

Input parameters	
bucket	gs://ai-dataset/HYP/Alvin/example_kubeflow
model-file	mnist_model.h5

Output parameters	
mnist-training-container-outputs	gs://ai-dataset/HYP/Alvin/example_kubeflow/mnist_model.h5

# Inspect the Experiments Orchestration

nodeflux

- We can inspect the details of each run
  - Input parameters
  - Output value
  - Logs

```
mnist-pipeline-r8cwf-2917520262

Artifacts  Input/Output  Volumes  Manifest  Logs

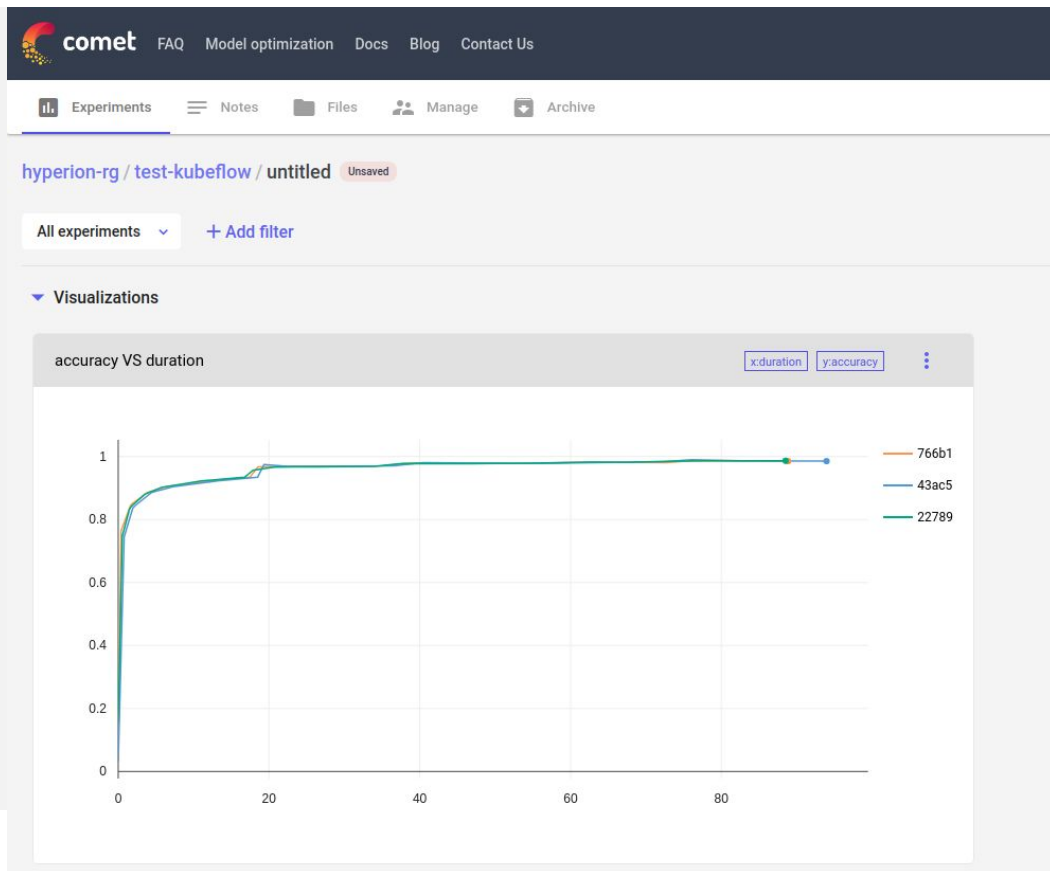
25 Train on 60000 samples, validate on 10000 samples
26 COMET INFO: Ignoring automatic log_parameter('verbose') because 'keras:verbose' is in COMET_LOGGING_PARAMETERS_IGNORE
27 COMET INFO: Ignoring automatic log_parameter('do_validation') because 'keras:do_validation' is in COMET_LOGGING_PARAMETERS_IGNORE
28 2020-01-29 11:30:26.428364: W tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could not load dynamic library 'libcuda.so.1'; dlderror:
29 2020-01-29 11:30:26.428574: E tensorflow/stream_executor/cuda/cuda_driver.cc:318] failed call to cuInit: UNKNOWN ERROR (303)
30 2020-01-29 11:30:26.428685: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be running on this host (mn
31 2020-01-29 11:30:26.430720: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not
32 2020-01-29 11:30:26.437690: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 2592020000 Hz
33 2020-01-29 11:30:26.438082: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x4591750 initialized for platform Host (this does not gu
34 2020-01-29 11:30:26.438197: I tensorflow/compiler/xla/service/service.cc:176] StreamExecutor device (0): Host, Default Version
35 Epoch 1/5
36 COMET INFO: Ignoring automatic log_metric('batch_batch') because 'keras:batch_batch' is in COMET_LOGGING_METRICS_IGNORE
37 COMET INFO: Ignoring automatic log_metric('batch_size') because 'keras:batch_size' is in COMET_LOGGING_METRICS_IGNORE
38 32/60000 [.....] - ETA: 2:03 - loss: 2.3343 - acc: 0.1562
39 Epoch 2/5
40 32/60000 [.....] - ETA: 10s - loss: 0.0499 - acc: 1.0000
41 Epoch 3/5
42 32/60000 [.....] - ETA: 10s - loss: 0.2928 - acc: 0.9062
43 Epoch 4/5
44 32/60000 [.....] - ETA: 15s - loss: 0.0241 - acc: 1.0000
45 Epoch 5/5
46 32/60000 [.....] - ETA: 13s - loss: 0.1521 - acc: 0.9688
47 WARNING:tensorflow:From /app/app.py:61: The name tf.gfile.Exists is deprecated. Please use tf.io.gfile.exists instead.
48
49 WARNING:tensorflow:From /app/app.py:62: The name tf.gfile.Remove is deprecated. Please use tf.io.gfile.remove instead.
50
51 WARNING:tensorflow:From /app/app.py:64: The name tf.gfile.Copy is deprecated. Please use tf.io.gfile.copy instead.
52
53 COMET INFO: -----
54 COMET INFO: Comet.ml Experiment Summary:
55 COMET INFO: Data:
56 COMET INFO: url: https://www.comet.ml/hyperion-rg/test-kubeflow/dd9e6d7fcdcf84467b88b90e3b65d1c2e
57 COMET INFO: Metrics [count] (min, max):
58 COMET INFO: acc [5] : (0.9368499979865491, 0.9856666922569275)
59 COMET INFO: accuracy [9375] : (0.15625, 1.0)
60 COMET INFO: batch_acc [940] : (0.15625, 1.0)
61 COMET INFO: batch_loss [940] : (0.009336319615598768, 2.3342535495758057)
62 COMET INFO: epoch_duration [5] : (11.286147636999958, 12.466364700001577)
63 COMET INFO: loss [9380] : (9.148869867203757e-05, 2.3342535495758057)
64 COMET INFO: step : 10940
65 COMET INFO: val_acc [5] : (0.9684000015258789, 0.9805999994727954)
66 COMET INFO: val_loss [5] : (0.06202604080243036, 0.10955566595941782)
67 COMET INFO: validate_batch_acc [160] : (0.9375, 1.0)
68 COMET INFO: validate_batch_loss [160] : (0.001953410916030407, 0.48382048444747925)
69 COMET INFO: Other [count]:
70 COMET INFO: trainable_params: 407050
71 COMET INFO: -----
72 COMET INFO: Uploading stats to Comet before program termination (may take several seconds)
```

# The Observer



# Comet.ml?

- A freemium framework to track code, experiments, and results
- Easy to integrate with famous deep learning framework
- Enable collaboration with our teams
- Visualization across different experiments



## Code Integration

- Kubeflow wrap script interaction inside a container as a python function, we call it ContainerOp
- We can compile the operations and pipeline as a single deployment file ( usually ends with .tar.gz ) and freely distributed to others so that they can use it

```
home > nodeflux > Documents > nodeflux-repo > research > MiniKF > mnist_training > app.py > ...
1  from comet_ml import Experiment
2  import argparse
3  from datetime import datetime
4  import tensorflow as tf
5  import os
6
7  os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = 'AI-training-a9aad66abc4e.json'
8
9  experiment = Experiment(api_key="0kZ5vwvCG7xBToH0eptFxtxu",
10                        project_name="test-kubeflow", workspace="hyperion-rg")
```

```
39  class CometMLCallback(tf.keras.callbacks.Callback):
40      def on_train_batch_end(self, batch, logs=None):
41          experiment.log_metric("loss", logs['loss'])
42          experiment.log_metric("accuracy", logs['acc'])
43
44
45  callbacks = [
46      CometMLCallback(),
47      # Interrupt training if val_loss stops improving for over 2 epochs
48      tf.keras.callbacks.EarlyStopping(patience=2, monitor='val_loss'),
49  ]
50
51  model.fit(x_train, y_train, batch_size=32, epochs=5, callbacks=callbacks,
52          validation_data=(x_test, y_test))
```

# Inspect the Experiments Progress

nodeflux

- Experiment link will spawn on the stdout log
- Will direct you to the comet.ml experiments page

Retry Clone run Terminate Archive



mnist-pipeline-r8cwf-2917520262

Artifacts Input/Output Volumes Manifest Logs

```
1 COMET INFO: Experiment is live on comet.ml https://www.comet.ml/hyperion-rg/test-kubeflow/dd9e6d7fcdf84467b08b90e3b65d1cce
2
3 WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/resource_variable_ops.py:1630: calling BaseResourceVariable.
4 Instructions for updating:
5 If using Keras pass *_constraint arguments to layers.
6 Model: "sequential"
```

# Inspect the Experiments Progress

nodeflux

- Similarly it will record logs, parameters, but it will visualize the metric graph over the training time

hyperion-rg / test-kubeflow / Experiment (dd9e6d7fcd84467b08b90e3b65d1cce)

Stop

Reproduce





# The Tracker



# Version Control for Data

- Download data as easy as cloning repository
- Maintain data version and changes
- Easy integration with Google Cloud Storage
- Avoid data corruption during download ( maintain data hashing )

## DVC tracks ML models and data sets

DVC is built to make ML models shareable and reproducible. It is designed to handle large files, data sets, machine learning models, and metrics as well as code.



```
$ dvc add images
```

```
$ dvc remote add -d myrepo s3://mybucket
```

```
$ dvc push
```

```
$ git add images.dvc
```

```
$ git commit
```

```
$ git push origin master
```

## Dataset with Git Integration

- Use it like you develop your code
- Integrate with git

# Clone it and Voila !

- As easy as clone a repo, select a version, pull

```
$ git clone git@gitlab.com:myrepo/data.git
```

```
$ git checkout v1
```

```
$ dvc pull
```

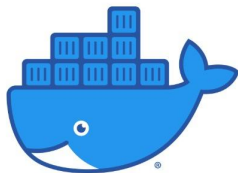
Machine Learning Toolkit

Orchestrator

Container

Training Script

Dataset Version and Tracking



kubernetes



nodeflux

# Summary



nodeflux

**Thank You**

[www.nodeflux.io](http://www.nodeflux.io)