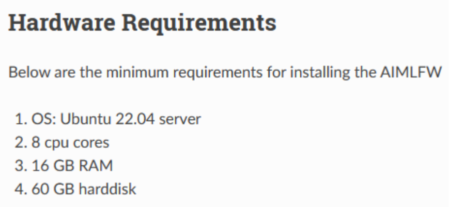
OSC AI/ML Framework(Release H) Install Notes

Tags	技術 概念
№ ID	WD-37
Last edited time	@October 19, 2023 11:56 PM

Hardware requirements

- Official hardware requirements



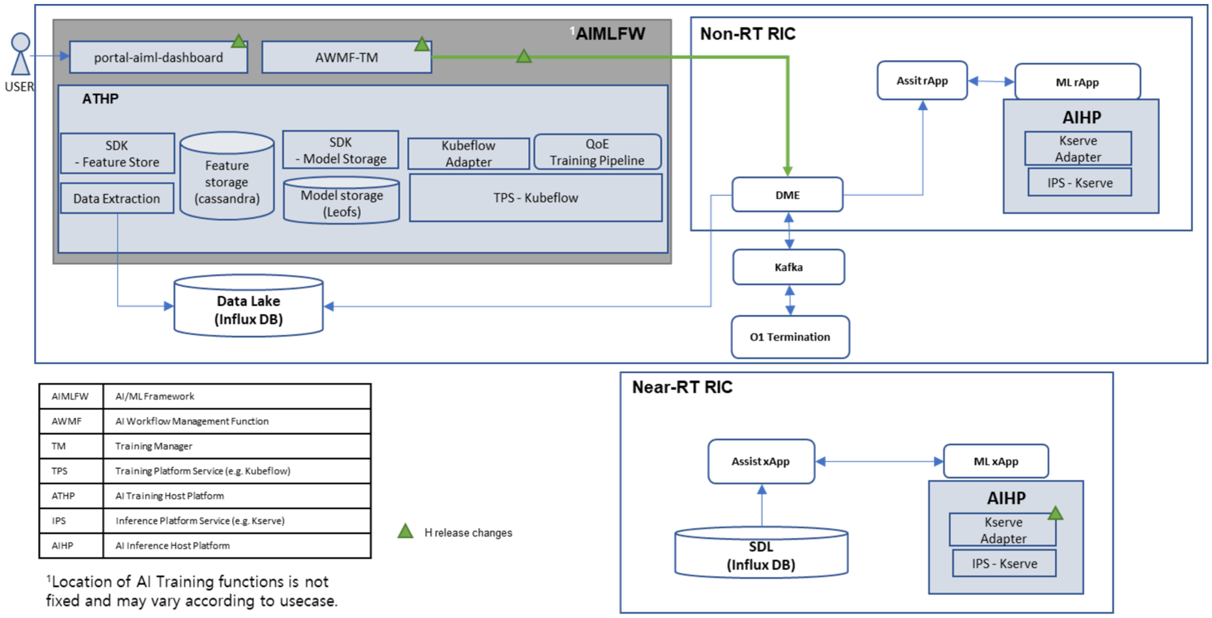
Hardware Requirements

Below are the minimum requirements for installing the AIMLFW

1. OS: Ubuntu 22.04 server
2. 8 cpu cores
3. 16 GB RAM
4. 60 GB harddisk

- (Option) Due to the insufficient disk found during the installation process, the configuration hardware resources are increased
  - RAM: UP to 24 GB
  - Hard disk: UP to 100 GB

AIMLFW(Release H) design diagram



AIMLFW	AI/ML Framework
AWMF	AI Workflow Management Function
TM	Training Manager
TPS	Training Platform Service (e.g. Kubeflow)
ATHP	AI Training Host Platform
IPS	Inference Platform Service (e.g. Kserve)
AIHP	AI Inference Host Platform

[1]Location of AI Training functions is not fixed and may vary according to usecase.

▼ (Optional) Create a virtual environment

- 1.Check **Python** version

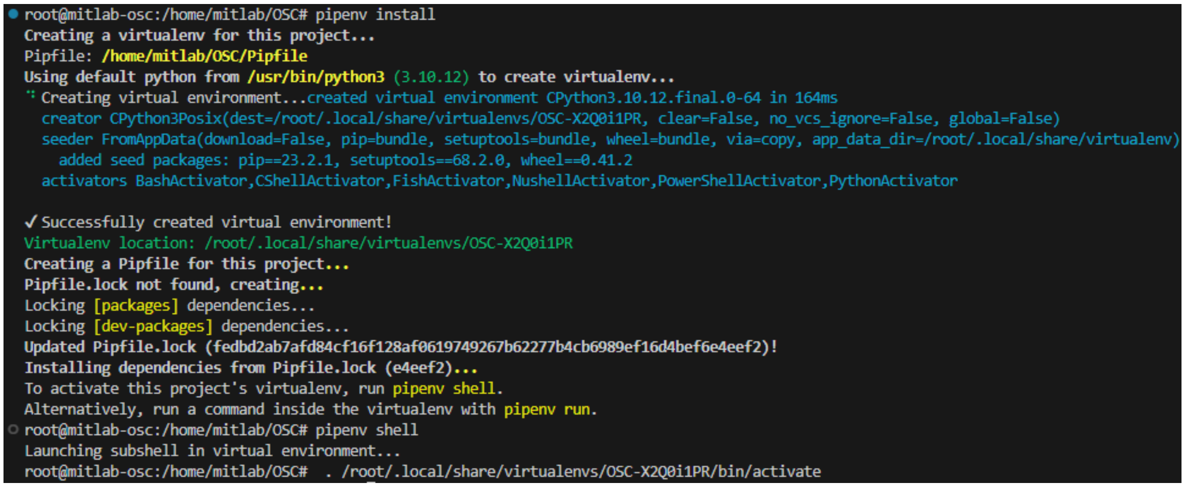
```
python --version
```

- 2. If not install `pipenv`

```
pip install pipenv
```

- 3. Make a directory,and then create a python virtual enviroment

```
mkdir project_name
cd project_name
pipenv install
```



```
● root@mitlab-osc:/home/mitlab/OSC# pipenv install
  Creating a virtualenv for this project...
  Pipfile: /home/mitlab/OSC/Pipfile
  Using default python from /usr/bin/python3 (3.10.12) to create virtualenv...
  ⠋ Creating virtual environment...created virtual environment CPython3.10.12.final.0-64 in 164ms
    creator CPython3Posix(dest=/root/.local/share/virtualenvs/OSC-X2Q0i1PR, clear=False, no_vcs_ignore=False, global=False)
    seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/root/.local/share/virtualenv)
      added seed packages: pip==23.2.1, setuptools==68.2.0, wheel==0.41.2
    activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

  ✓ Successfully created virtual environment!
  Virtualenv location: /root/.local/share/virtualenvs/OSC-X2Q0i1PR
  Creating a Pipfile for this project...
  Pipfile.lock not found, creating...
  Locking [packages] dependencies...
  Locking [dev-packages] dependencies...
  Updated Pipfile.lock (fedbd2ab7afd84cf16f128af0619749267b62277b4cb6989ef16d4bef6e4eef2)!
  Installing dependencies from Pipfile.lock (e4eef2)...
  To activate this project's virtualenv, run pipenv shell.
  Alternatively, run a command inside the virtualenv with pipenv run.
○ root@mitlab-osc:/home/mitlab/OSC# pipenv shell
  Launching subshell in virtual environment...
  root@mitlab-osc:/home/mitlab/OSC#  . /root/.local/share/virtualenvs/OSC-X2Q0i1PR/bin/activate
```

- Activate python virtual enviroment

```
pipenv shell
```

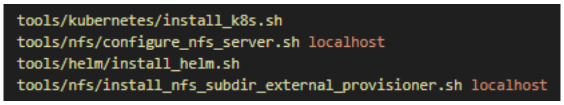
Step 1. Software installation and deployment

1-1. Download aimlfw file

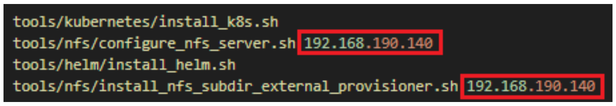
```
git clone "https://gerrit.o-ran-sc.org/r/aiml-fw/aimlfw-dep"
cd aimlfw-dep
```

1-2 Revise install_traininghost.sh

- Replace localhost to **<ip_address>**



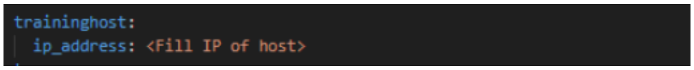
```
tools/kubernetes/install_k8s.sh
tools/nfs/configure_nfs_server.sh localhost
tools/helm/install_helm.sh
tools/nfs/install_nfs_subdir_external_provisioner.sh localhost
```



```
tools/kubernetes/install_k8s.sh
tools/nfs/configure_nfs_server.sh 192.168.190.140
tools/helm/install_helm.sh
tools/nfs/install_nfs_subdir_external_provisioner.sh 192.168.190.140
```

1-3. Updated RECIPE_EXAMPLE/example_recipe_latest_stable.yaml

- Fill host IP : **<traininghost ip_address>**



```
traininghost:
  ip_address: <Fill IP of host>
```

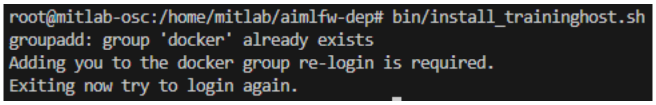
For example:

```
traininghost:
  ip_address: 192.168.190.140
```

1-4. Run install_traininghost.sh

- Install traininghost

```
bin/install_traininghost.sh
```

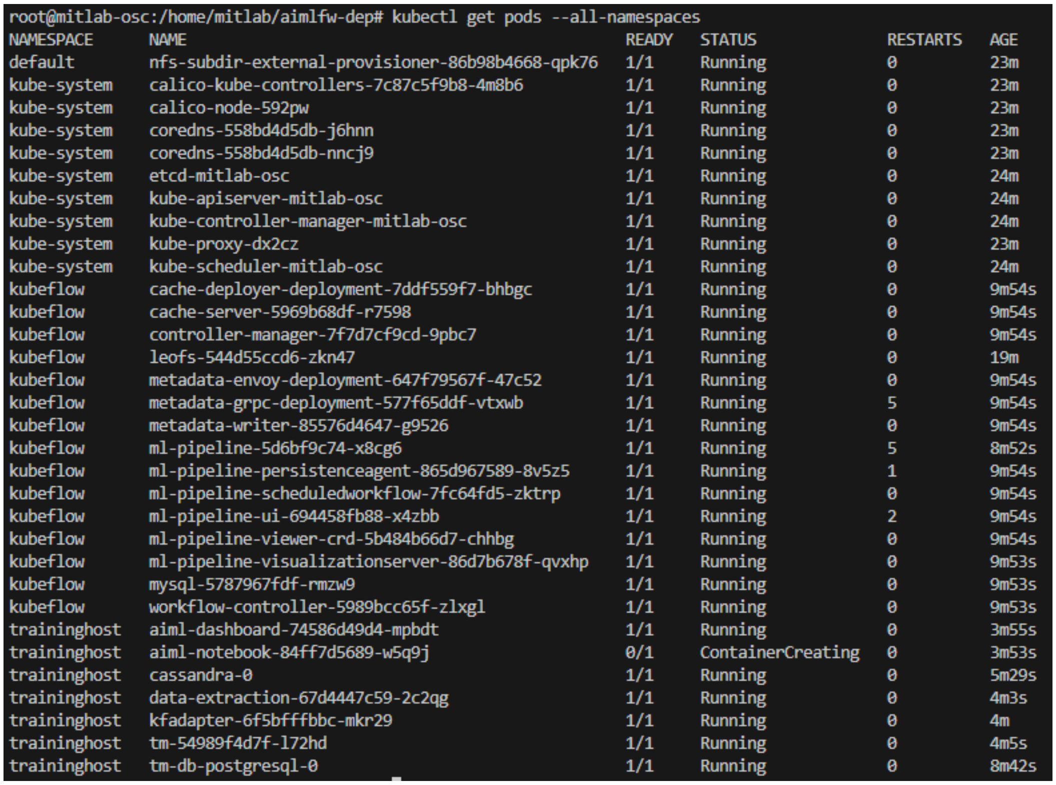


```
root@mitlab-osc:/home/mitlab/aimlfw-dep# bin/install_traininghost.sh
groupadd: group 'docker' already exists
Adding you to the docker group re-login is required.
Exiting now try to login again.
```

Re-login，command `bin/install_traininghost.sh`

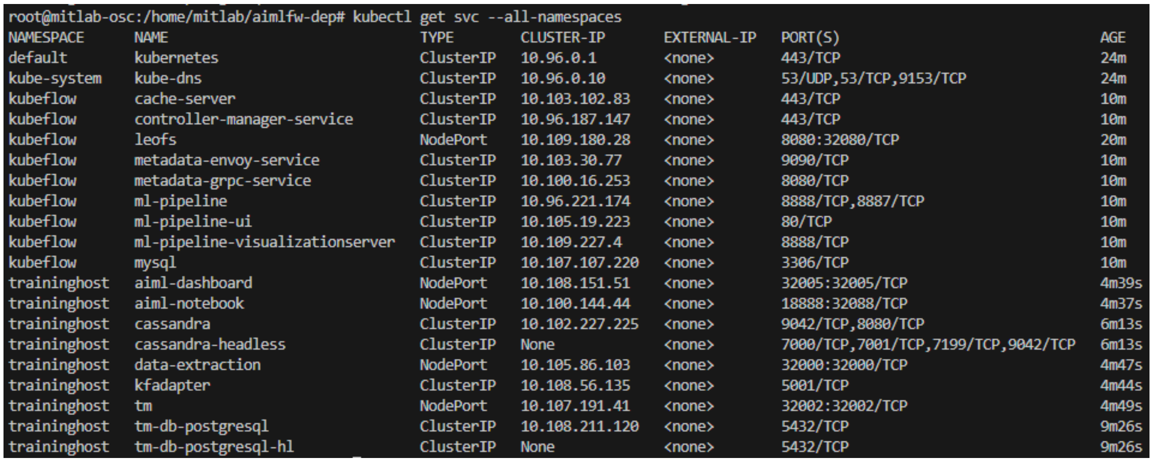
- After you complete installation, you may see the figure like this.

```
kubectl get pods --all-namespaces
```



```
root@mitlab-osc:/home/mitlab/aimlfw-dep# kubectl get pods --all-namespaces
NAMESPACE      NAME                                             READY   STATUS              RESTARTS   AGE
default        nfs-subdir-external-provisioner-86b98b4668-qpk76   1/1     Running             0          23m
kube-system    calico-kube-controllers-7c87c5f9b8-4m8b6           1/1     Running             0          23m
kube-system    calico-node-592pw                                  1/1     Running             0          23m
kube-system    coredns-558bd4d5db-j6hnn                           1/1     Running             0          23m
kube-system    coredns-558bd4d5db-nncj9                           1/1     Running             0          23m
kube-system    etcd-mitlab-osc                                    1/1     Running             0          24m
kube-system    kube-apiserver-mitlab-osc                          1/1     Running             0          24m
kube-system    kube-controller-manager-mitlab-osc                 1/1     Running             0          24m
kube-system    kube-proxy-dx2cz                                   1/1     Running             0          23m
kube-system    kube-scheduler-mitlab-osc                          1/1     Running             0          24m
kubeflow       cache-deployer-deployment-7ddf559f7-bhbgc          1/1     Running             0          9m54s
kubeflow       cache-server-5969b68df-r7598                       1/1     Running             0          9m54s
kubeflow       controller-manager-7f7d7cf9cd-9pbc7                1/1     Running             0          9m54s
kubeflow       leofs-544d55ccd6-zkn47                             1/1     Running             0          19m
kubeflow       metadata-envoy-deployment-647f79567f-47c52         1/1     Running             0          9m54s
kubeflow       metadata-grpc-deployment-577f65ddf-vtxwb           1/1     Running             5          9m54s
kubeflow       metadata-writer-85576d4647-g9526                   1/1     Running             0          9m54s
kubeflow       ml-pipeline-5d6bf9c74-x8cg6                        1/1     Running             5          8m52s
kubeflow       ml-pipeline-persistenceagent-865d967589-8v5z5      1/1     Running             1          9m54s
kubeflow       ml-pipeline-scheduledworkflow-7fc64fd5-zktrp       1/1     Running             0          9m54s
kubeflow       ml-pipeline-ui-694458fb88-x4zbb                    1/1     Running             2          9m54s
kubeflow       ml-pipeline-viewer-crd-5b484b66d7-chhbg            1/1     Running             0          9m54s
kubeflow       ml-pipeline-visualizationserver-86d7b678f-qvxhp    1/1     Running             0          9m53s
kubeflow       mysql-5787967fdf-rmzw9                             1/1     Running             0          9m53s
kubeflow       workflow-controller-5989bcc65f-zlxgl               1/1     Running             0          9m53s
traininghost   aiml-dashboard-74586d49d4-mpbdt                    1/1     Running             0          3m55s
traininghost   aiml-notebook-84ff7d5689-w5q9j                     0/1     ContainerCreating   0          3m53s
traininghost   cassandra-0                                        1/1     Running             0          5m29s
traininghost   data-extraction-67d4447c59-2c2qg                   1/1     Running             0          4m3s
traininghost   kfadapter-6f5bfffbbc-mkr29                         1/1     Running             0          4m
traininghost   tm-54989f4d7f-l72hd                                1/1     Running             0          4m5s
traininghost   tm-db-postgresql-0                                 1/1     Running             0          8m42s
```

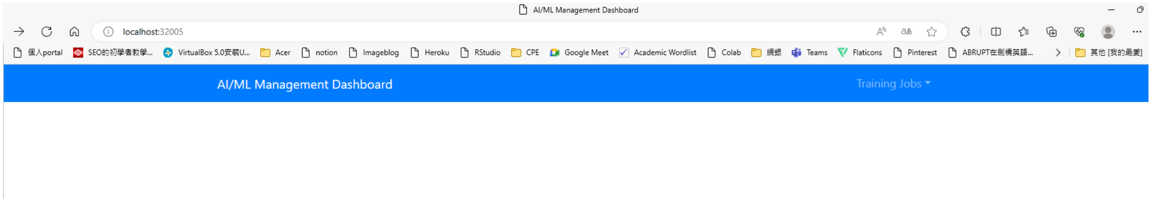
```
kubectl get svc --all-namespaces
```



```
root@mitlab-osc:/home/mitlab/aimlfw-dep# kubectl get svc --all-namespaces
NAMESPACE      NAME                              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                                  AGE
default        kubernetes                        ClusterIP   10.96.0.1        <none>        443/TCP                                  24m
kube-system    kube-dns                          ClusterIP   10.96.0.10       <none>        53/UDP,53/TCP,9153/TCP                   24m
kubeflow       cache-server                      ClusterIP   10.103.102.83    <none>        443/TCP                                  10m
kubeflow       controller-manager-service        ClusterIP   10.96.187.147    <none>        443/TCP                                  10m
kubeflow       leofs                             NodePort    10.109.180.28    <none>        8080:32080/TCP                           20m
kubeflow       metadata-envoy-service            ClusterIP   10.103.30.77     <none>        9090/TCP                                 10m
kubeflow       metadata-grpc-service             ClusterIP   10.100.16.253    <none>        8080/TCP                                 10m
kubeflow       ml-pipeline                       ClusterIP   10.96.221.174    <none>        8888/TCP,8887/TCP                        10m
kubeflow       ml-pipeline-ui                    ClusterIP   10.105.19.223    <none>        80/TCP                                   10m
kubeflow       ml-pipeline-visualizationserver   ClusterIP   10.109.227.4     <none>        8888/TCP                                 10m
kubeflow       mysql                             ClusterIP   10.107.107.220   <none>        3306/TCP                                 10m
traininghost   aiml-dashboard                    NodePort    10.108.151.51    <none>        32005:32005/TCP                          4m39s
traininghost   aiml-notebook                     NodePort    10.100.144.44    <none>        18888:32088/TCP                          4m37s
traininghost   cassandra                         ClusterIP   10.102.227.225   <none>        9042/TCP,8080/TCP                        6m13s
traininghost   cassandra-headless                ClusterIP   None             <none>        7000/TCP,7001/TCP,7199/TCP,9042/TCP      6m13s
traininghost   data-extraction                   NodePort    10.105.86.103    <none>        32000:32000/TCP                          4m47s
traininghost   kfadapter                         ClusterIP   10.108.56.135    <none>        5001/TCP                                 4m44s
traininghost   tm                                NodePort    10.107.191.41    <none>        32002:32002/TCP                          4m49s
traininghost   tm-db-postgresql                  ClusterIP   10.108.211.120   <none>        5432/TCP                                 9m26s
traininghost   tm-db-postgresql-hl               ClusterIP   None             <none>        5432/TCP                                 9m26s
```

- Check the AIMLFW dashboard by using the following url,remember to do the port forwarding if you use VM.

```
http://<Your VM IP>:32005/
```



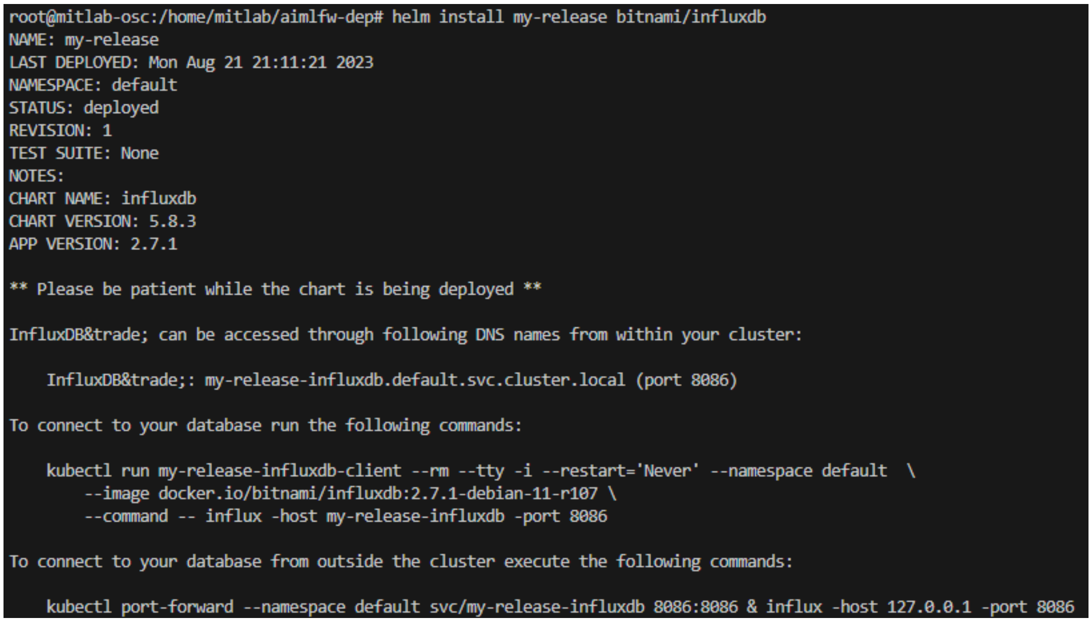
▼ Step 2. Install Influx DB as datalake

(Pre-Checking) Given that the OSC's AI/ML Framework already assumes that Influx DB has been installed as the Datalake, if you haven't yet installed the Datalake (InfluxDB), please proceed with the installation of Influx DB first.

▼ 2-1. Install Influx DB and create bucket

- Install Influx DB

```
helm repo add bitnami https://charts.bitnami.com/bitnami
helm install my-release bitnami/influxdb
```

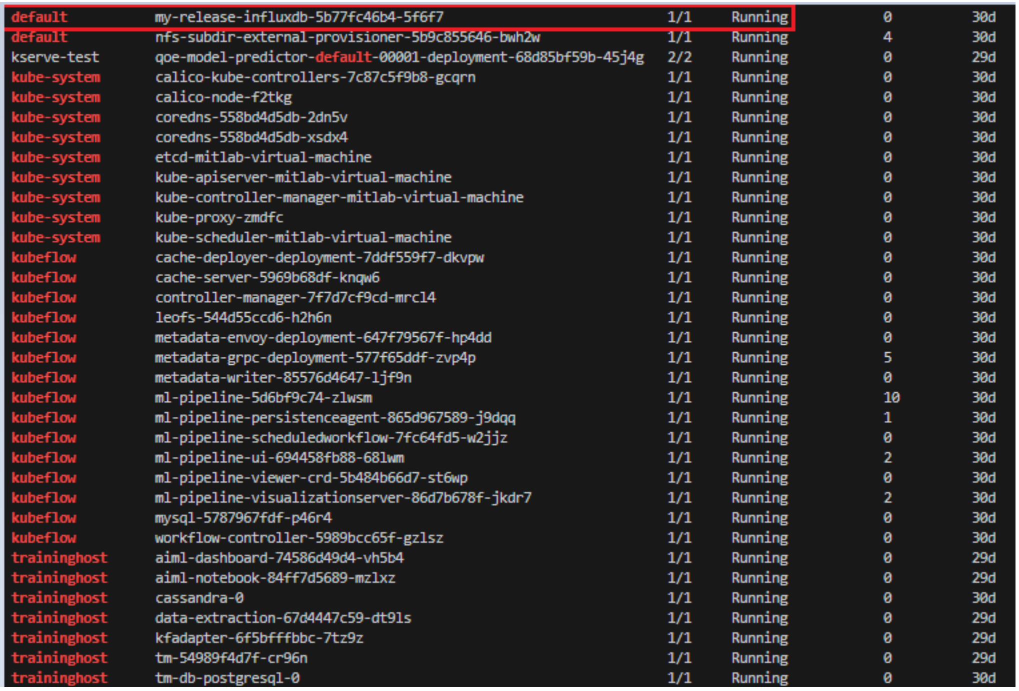


```
root@mitlab-osc:/home/mitlab/aimlfw-dep# helm install my-release bitnami/influxdb
NAME: my-release
LAST DEPLOYED: Mon Aug 21 21:11:21 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: influxdb
CHART VERSION: 5.8.3
APP VERSION: 2.7.1

** Please be patient while the chart is being deployed **

InfluxDB&trade; can be accessed through following DNS names from within your cluster:

    InfluxDB&trade;: my-release-influxdb.default.svc.cluster.local (port 8086)

To connect to your database run the following commands:

    kubectl run my-release-influxdb-client --rm --tty -i --restart='Never' --namespace default  \
        --image docker.io/bitnami/influxdb:2.7.1-debian-11-r107 \
        --command -- influx -host my-release-influxdb -port 8086

To connect to your database from outside the cluster execute the following commands:

    kubectl port-forward --namespace default svc/my-release-influxdb 8086:8086 & influx -host 127.0.0.1 -port 8086
```

show helm install my-release bitnami/influxdb

- Use this command to find influxdb pod.

```
kubectl get pods -A
```



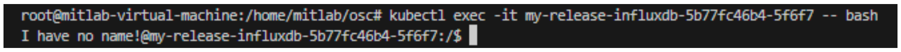
```
default          my-release-influxdb-5b77fc46b4-5f6f7                          1/1     Running     0       30d
default          nfs-subdir-external-provisioner-5b9c855646-bwh2w              1/1     Running     4       30d
kserve-test      qoe-model-predictor-default-00001-deployment-68d85bf59b-45j4g 2/2     Running     0       29d
kube-system      calico-kube-controllers-7c87c5f9b8-gcqrn                      1/1     Running     0       30d
kube-system      calico-node-f2tkg                                             1/1     Running     0       30d
kube-system      coredns-558bd4d5db-2dn5v                                      1/1     Running     0       30d
kube-system      coredns-558bd4d5db-xsdx4                                      1/1     Running     0       30d
kube-system      etcd-mitlab-virtual-machine                                   1/1     Running     0       30d
kube-system      kube-apiserver-mitlab-virtual-machine                         1/1     Running     0       30d
kube-system      kube-controller-manager-mitlab-virtual-machine                1/1     Running     0       30d
kube-system      kube-proxy-zmdfc                                              1/1     Running     0       30d
kube-system      kube-scheduler-mitlab-virtual-machine                         1/1     Running     0       30d
kubeflow         cache-deployer-deployment-7ddf559f7-dkvpw                     1/1     Running     0       30d
kubeflow         cache-server-5969b68df-knqw6                                  1/1     Running     0       30d
kubeflow         controller-manager-7f7d7cf9cd-mrcl4                           1/1     Running     0       30d
kubeflow         leofs-544d55ccd6-h2h6n                                        1/1     Running     0       30d
kubeflow         metadata-envoy-deployment-647f79567f-hp4dd                    1/1     Running     0       30d
kubeflow         metadata-grpc-deployment-577f65ddf-zvp4p                      1/1     Running     5       30d
kubeflow         metadata-writer-85576d4647-ljf9n                              1/1     Running     0       30d
kubeflow         ml-pipeline-5d6bf9c74-zlwsm                                   1/1     Running     10      30d
kubeflow         ml-pipeline-persistenceagent-865d967589-j9dqq                 1/1     Running     1       30d
kubeflow         ml-pipeline-scheduledworkflow-7fc64fd5-w2jjz                  1/1     Running     0       30d
kubeflow         ml-pipeline-ui-694458fb88-68lwm                               1/1     Running     2       30d
kubeflow         ml-pipeline-viewer-crd-5b484b66d7-st6wp                       1/1     Running     0       30d
kubeflow         ml-pipeline-visualizationserver-86d7b678f-jkdr7               1/1     Running     2       30d
kubeflow         mysql-5787967fdf-p46r4                                        1/1     Running     0       30d
kubeflow         workflow-controller-5989bcc65f-gzlsz                          1/1     Running     0       30d
traininghost     aiml-dashboard-74586d49d4-vh5b4                               1/1     Running     0       29d
traininghost     aiml-notebook-84ff7d5689-mzlxz                                1/1     Running     0       29d
traininghost     cassandra-0                                                   1/1     Running     0       30d
traininghost     data-extraction-67d4447c59-dt9ls                              1/1     Running     0       29d
traininghost     kfadapter-6f5bfffbbc-7tz9z                                    1/1     Running     0       29d
traininghost     tm-54989f4d7f-cr96n                                           1/1     Running     0       29d
traininghost     tm-db-postgresql-0                                            1/1     Running     0       30d
```

- After you find, use this command to get into the pod.

```
kubectl exec -it <pod name> -- bash
```

For example：

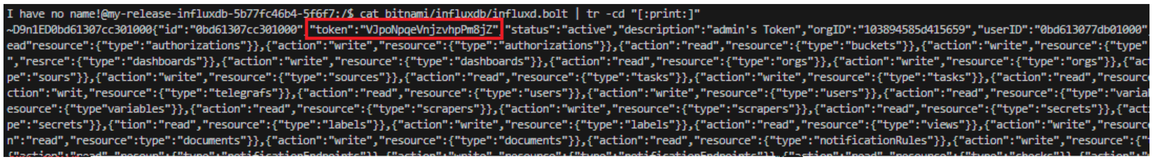
```
kubectl exec -it my-release-influxdb-5b77fc46b4-5f6f7 -- bash
```



```
root@mitlab-virtual-machine:/home/mitlab/osc# kubectl exec -it my-release-influxdb-5b77fc46b4-5f6f7 -- bash
I have no name!@my-release-influxdb-5b77fc46b4-5f6f7:/$
```

- From below command we can get username, org name, org id and access token

```
cat bitnami/influxdb/influxd.bolt | tr -cd "[:print:]"
```



```
I have no name!@my-release-influxdb-5b77fc46b4-5f6f7:/$ cat bitnami/influxdb/influxd.bolt | tr -cd "[:print:]"
~D9n1ED0bd61307cc301000{"id":"0bd61307cc301000","token":"VJpoNpqeVnjzvhpPm8jZ","status":"active","description":"admin's Token","orgID":"103894585d415659","userID":"0bd613077db01000"
ead"resource":{"type":"authorizations"}},{"action":"write","resource":{"type":"authorizations"}},{"action":"read","resource":{"type":"buckets"}},{"action":"write","resource":{"type"
","resrce":{"type":"dashboards"}},{"action":"write","resource":{"type":"dashboards"}},{"action":"read","resource":{"type":"orgs"}},{"action":"write","resource":{"type":"orgs"}},{"ac
pe":"sours"}},{"action":"write","resource":{"type":"sources"}},{"action":"read","resource":{"type":"tasks"}},{"action":"write","resource":{"type":"tasks"}},{"action":"read","resourc
ction":"writ,"resource":{"type":"telegrafs"}},{"action":"read","resource":{"type":"users"}},{"action":"write","resource":{"type":"users"}},{"action":"read","resource":{"type":"varia
esource":{"type"variables"}},{"action":"read","resource":{"type":"scrapers"}},{"action":"write","resource":{"type":"scrapers"}},{"action":"read","resource":{"type":"secrets"}},{"act
pe":"secrets"}},{"tion":"read","resource":{"type":"labels"}},{"action":"write","resource":{"type":"labels"}},{"action":"read","resource":{"type":"views"}},{"action":"write","resourc
n":"read","resource":type":"documents"}},{"action":"write","resource":{"type":"documents"}},{"action":"read","resource":{"type":"notificationRules"}},{"action":"write","resource":{"
{"action":"read","resour":{"type":"notificationEndpoints"}},{"action":"write","resource":{"type":"notificationEndpoints"}},{"action":"read","resource":{"type":"checks"}},{"action":"
```

token: "VJpoNpqeVnjzvhpPm8jZ”

- Execute below from inside Influx DB container to create a bucket

```
influx bucket create -n UEData -o primary -t <token>
```

For example：

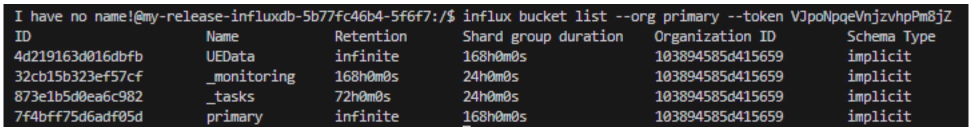
```
influx bucket create -n UEData -o primary -t VJpoNpqeVnjzvhpPm8jZ
```

- You can check bucket lists by this command

```
influx bucket list --org <org_name> --token <API_Token>
```

For example：

```
influx bucket list --org primary --token VJpoNpqeVnjzvhpPm8jZ
```



```
I have no name!@my-release-influxdb-5b77fc46b4-5f6f7:/$ influx bucket list --org primary --token VJpoNpqeVnjzvhpPm8jZ
ID                      Name            Retention       Shard group duration    Organization ID         Schema Type
4d219163d016dbfb        UEData          infinite        168h0m0s                103894585d415659        implicit
32cb15b323ef57cf        _monitoring     168h0m0s        24h0m0s                 103894585d415659        implicit
873e1b5d0ea6c982        _tasks          72h0m0s         24h0m0s                 103894585d415659        implicit
7f4bff75d6adf05d        primary         infinite        168h0m0s                103894585d415659        implicit
```

▼ 2-2. Update recipe file `RECIPE_EXAMPLE/example_recipe_latest_stable.yaml`

- Update recipe file `RECIPE_EXAMPLE/example_recipe_latest_stable.yaml` which includes update of VM IP and datalake details.

```
vim RECIPE_EXAMPLE/example_recipe_latest_stable.yaml
```

change IP of **traininghost、datalake.influxdb**

```
traininghost:
  ip_address: 192.168.190.140


datalake:
  influxdb:
    host: 192.168.190.140
    port: 8086
    orgname: primary
    bucket: UEData
    token: VJpoNpqeVnjzvhpPm8jZ
```

- Once updated, follow the below steps for reinstall of some components

```
bin/uninstall.sh
bin/install.sh -f RECIPE_EXAMPLE/example_recipe_latest_stable.yaml
```

▼ 2-3. Accessing applications in the cluster using port forwarding to send data.

- Install the following dependencies

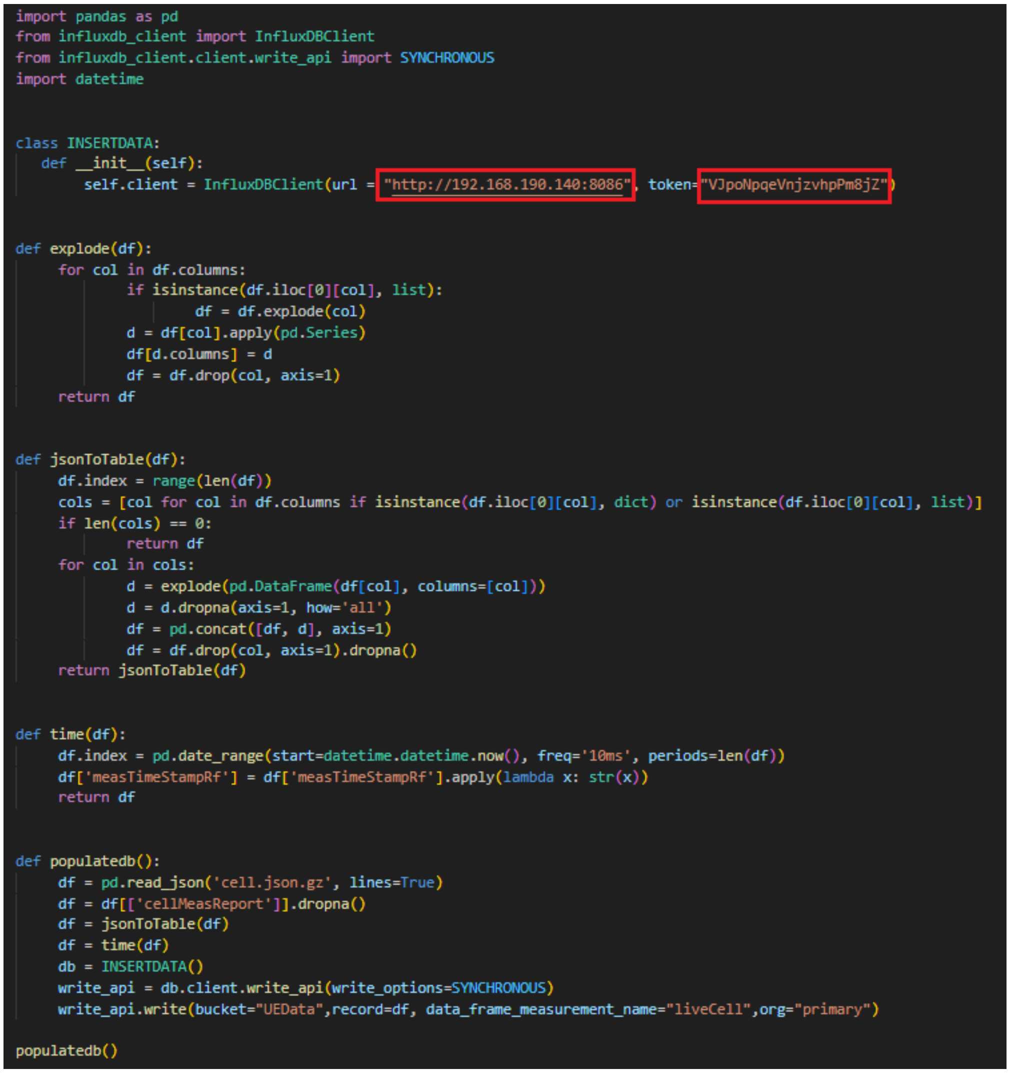
```
sudo apt-get install python3-pip
sudo pip3 install pandas
sudo pip3 install influxdb_client
```

- Use the `insert.py` in `ric-app/qp repository` to upload the qoe data in Influx DB

```
git clone -b f-release https://gerrit.o-ran-sc.org/r/ric-app/qp
cd qp/qp
```

- Change **<localhost>** and Update **< token >** in `insert.py` file.

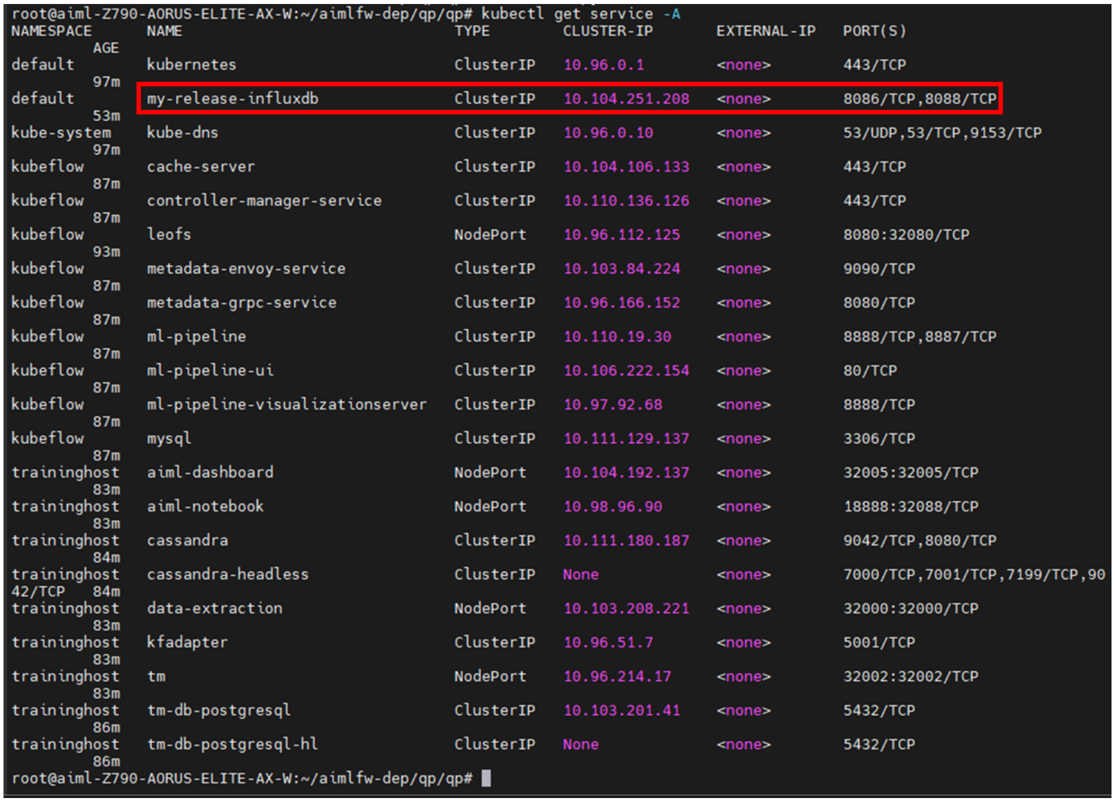
```
import pandas as pd
from influxdb_client import InfluxDBClient
from influxdb_client.client.write_api import SYNCHRONOUS
import datetime


class INSERTDATA:

   def __init__(self):
        self.client = InfluxDBClient(url = "http://localhost:8086", token="<token>")


def explode(df):
     for col in df.columns:
             if isinstance(df.iloc[0][col], list):
                     df = df.explode(col)
             d = df[col].apply(pd.Series)
             df[d.columns] = d
             df = df.drop(col, axis=1)
     return df


def jsonToTable(df):
     df.index = range(len(df))
     cols = [col for col in df.columns if isinstance(df.iloc[0][col], dict) or isinstance(df.iloc[0][col], list)]
     if len(cols) == 0:
             return df
     for col in cols:
             d = explode(pd.DataFrame(df[col], columns=[col]))
             d = d.dropna(axis=1, how='all')
             df = pd.concat([df, d], axis=1)
             df = df.drop(col, axis=1).dropna()
     return jsonToTable(df)


def time(df):
     df.index = pd.date_range(start=datetime.datetime.now(), freq='10ms', periods=len(df))
     df['measTimeStampRf'] = df['measTimeStampRf'].apply(lambda x: str(x))
     return df


def populatedb():
     df = pd.read_json('cell.json.gz', lines=True)
     df = df[['cellMeasReport']].dropna()
     df = jsonToTable(df)
     df = time(df)
     db = INSERTDATA()
     write_api = db.client.write_api(write_options=SYNCHRONOUS)
     write_api.write(bucket="UEData",record=df, data_frame_measurement_name="liveCell",org="primary")

populatedb()
```



```
import pandas as pd
from influxdb_client import InfluxDBClient
from influxdb_client.client.write_api import SYNCHRONOUS
import datetime


class INSERTDATA:
   def __init__(self):
        self.client = InfluxDBClient(url = "http://192.168.190.140:8086", token="VJpoNpqeVnjzvhpPm8jZ")


def explode(df):
     for col in df.columns:
             if isinstance(df.iloc[0][col], list):
                     df = df.explode(col)
             d = df[col].apply(pd.Series)
             df[d.columns] = d
             df = df.drop(col, axis=1)
     return df


def jsonToTable(df):
     df.index = range(len(df))
     cols = [col for col in df.columns if isinstance(df.iloc[0][col], dict) or isinstance(df.iloc[0][col], list)]
     if len(cols) == 0:
             return df
     for col in cols:
             d = explode(pd.DataFrame(df[col], columns=[col]))
             d = d.dropna(axis=1, how='all')
             df = pd.concat([df, d], axis=1)
             df = df.drop(col, axis=1).dropna()
     return jsonToTable(df)


def time(df):
     df.index = pd.date_range(start=datetime.datetime.now(), freq='10ms', periods=len(df))
     df['measTimeStampRf'] = df['measTimeStampRf'].apply(lambda x: str(x))
     return df


def populatedb():
     df = pd.read_json('cell.json.gz', lines=True)
     df = df[['cellMeasReport']].dropna()
     df = jsonToTable(df)
     df = time(df)
     db = INSERTDATA()
     write_api = db.client.write_api(write_options=SYNCHRONOUS)
     write_api.write(bucket="UEData",record=df, data_frame_measurement_name="liveCell",org="primary")

populatedb()
```

- Follow below command to port forward to access Influx DB
    - **Step 1：Check influx service name and port**

```
kubectl get service -A
```



```
root@aiml-Z790-AORUS-ELITE-AX-W:~/aimlfw-dep/qp/qp# kubectl get service -A
NAMESPACE      NAME                              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
         AGE
default        kubernetes                        ClusterIP   10.96.0.1        <none>        443/TCP
         97m
default        my-release-influxdb               ClusterIP   10.104.251.208   <none>        8086/TCP,8088/TCP
         53m
kube-system    kube-dns                          ClusterIP   10.96.0.10       <none>        53/UDP,53/TCP,9153/TCP
         97m
kubeflow       cache-server                      ClusterIP   10.104.106.133   <none>        443/TCP
         87m
kubeflow       controller-manager-service        ClusterIP   10.110.136.126   <none>        443/TCP
         87m
kubeflow       leofs                             NodePort    10.96.112.125    <none>        8080:32080/TCP
         93m
kubeflow       metadata-envoy-service            ClusterIP   10.103.84.224    <none>        9090/TCP
         87m
kubeflow       metadata-grpc-service             ClusterIP   10.96.166.152    <none>        8080/TCP
         87m
kubeflow       ml-pipeline                       ClusterIP   10.110.19.30     <none>        8888/TCP,8887/TCP
         87m
kubeflow       ml-pipeline-ui                    ClusterIP   10.106.222.154   <none>        80/TCP
         87m
kubeflow       ml-pipeline-visualizationserver   ClusterIP   10.97.92.68      <none>        8888/TCP
         87m
kubeflow       mysql                             ClusterIP   10.111.129.137   <none>        3306/TCP
         87m
traininghost   aiml-dashboard                    NodePort    10.104.192.137   <none>        32005:32005/TCP
         83m
traininghost   aiml-notebook                     NodePort    10.98.96.90      <none>        18888:32088/TCP
         83m
traininghost   cassandra                         ClusterIP   10.111.180.187   <none>        9042/TCP,8080/TCP
         84m
traininghost   cassandra-headless                ClusterIP   None             <none>        7000/TCP,7001/TCP,7199/TCP,90
42/TCP   84m
traininghost   data-extraction                   NodePort    10.103.208.221   <none>        32000:32000/TCP
         83m
traininghost   kfadapter                         ClusterIP   10.96.51.7       <none>        5001/TCP
         83m
traininghost   tm                                NodePort    10.96.214.17     <none>        32002:32002/TCP
         83m
traininghost   tm-db-postgresql                  ClusterIP   10.103.201.41    <none>        5432/TCP
         86m
traininghost   tm-db-postgresql-hl               ClusterIP   None             <none>        5432/TCP
         86m
root@aiml-Z790-AORUS-ELITE-AX-W:~/aimlfw-dep/qp/qp#
```

My influx service name：my-release-influxdb

My port：8086/TCP,8088/TCP

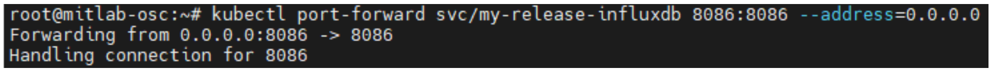
- **Step 2：Open new terminal and follow below command to port forward to Influx DB**

```
kubectl port-forward svc/<Your influxDB service name> 8086:<Your influxDB service port> --address=0.0.0.0
```

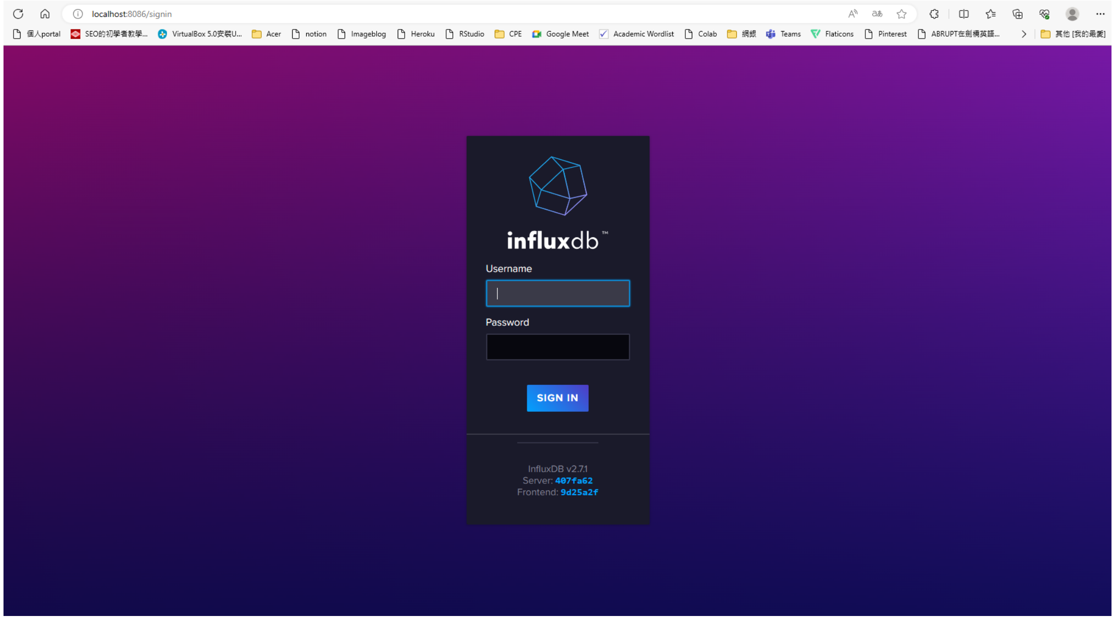
For example：

```
kubectl port-forward svc/my-release-influxdb 8086:8086 --address=0.0.0.0
```

If successful you will get this informaton in your new terminal.



```
root@mitlab-osc:~# kubectl port-forward svc/my-release-influxdb 8086:8086 --address=0.0.0.0
Forwarding from 0.0.0.0:8086 -> 8086
Handling connection for 8086
```



- **Step 3：Back to the terminal and run this command to insert data**

```
python3 insert.py
```

- To check inserted data in Influx DB, execute below command inside the Influx DB container:
- Step 1. Get into influxdb pod.

```
kubectl exec -it my-release-influxdb-5b77fc46b4-5f6f7 --bash
```

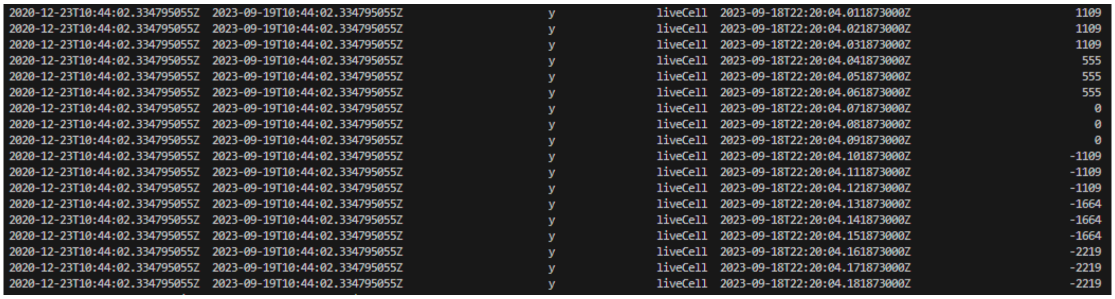
- Step 2. Check the data in the container.

```
influx query  'from(bucket: "UEData") |> range(start: -1000d)' -o primary -t <token>
```

For example:

```
influx query  'from(bucket: "UEData") |> range(start: -1000d)' -o primary -t VJpoNpqeVnjzvhpPm8jZ
```

and you will see the information like this figure.

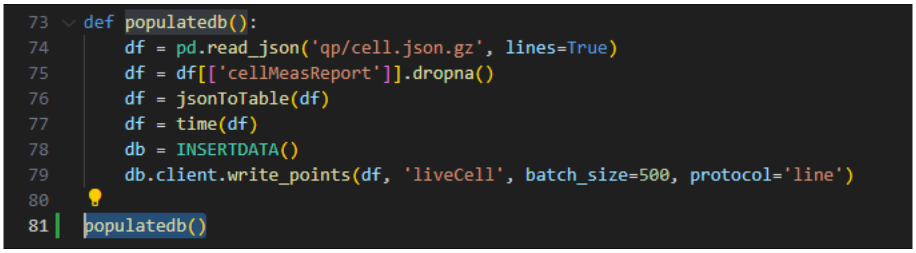


```
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.011873000Z                 1109
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.021873000Z                 1109
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.031873000Z                 1109
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.041873000Z                  555
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.051873000Z                  555
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.061873000Z                  555
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.071873000Z                    0
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.081873000Z                    0
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.091873000Z                    0
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.101873000Z                -1109
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.111873000Z                -1109
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.121873000Z                -1109
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.131873000Z                -1664
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.141873000Z                -1664
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.151873000Z                -1664
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.161873000Z                -2219
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.171873000Z                -2219
2020-12-23T10:44:02.334795055Z  2023-09-19T10:44:02.334795055Z                     y              liveCell  2023-09-18T22:20:04.181873000Z                -2219
```

▼ (Problem) After inserting data into Influx DB, querying the Influx DB data did not find the data.

A3-1.

The latest version of `insert.py` seems to be missing the call to `populatedb()`. After manually adding the `populatedb()` call, InfluxDB started to populate with data.



```
def populatedb():
    df = pd.read_json('qp/cell.json.gz', lines=True)
    df = df[['cellMeasReport']].dropna()
    df = jsonToTable(df)
    df = time(df)
    db = INSERTDATA()
    db.client.write_points(df, 'liveCell', batch_size=500, protocol='line')

populatedb()
```

A3-2.

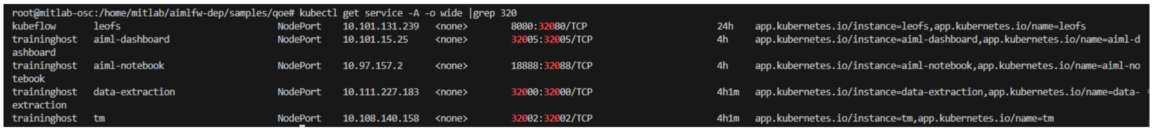
After **waiting for many hours**, the data appeared.

▼ Step 3. Create training function

▼ 3-1. Create training function

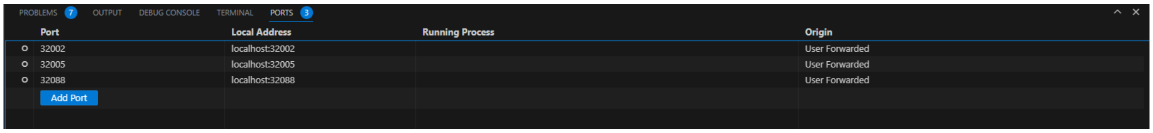
- Check the aiml-notebook service (port 32088)

```
kubectl get service -A -o wide |grep 320
```



```
root@mitlab-osc:/home/mitlab/aimlfw-dep/samples/qoe# kubectl get service -A -o wide |grep 320
kubeflow      leofs                   NodePort   10.101.131.239   <none>       8080:32080/TCP              24h    app.kubernetes.io/instance=leofs,app.kubernetes.io/name=leofs
traininghost  aiml-dashboard          NodePort   10.101.15.25     <none>       32005:32005/TCP             4h     app.kubernetes.io/instance=aiml-dashboard,app.kubernetes.io/name=aiml-d
ashboard
traininghost  aiml-notebook           NodePort   10.97.157.2      <none>       18888:32088/TCP             4h     app.kubernetes.io/instance=aiml-notebook,app.kubernetes.io/name=aiml-no
tebook
traininghost  data-extraction         NodePort   10.111.227.183   <none>       32000:32000/TCP             4h1m   app.kubernetes.io/instance=data-extraction,app.kubernetes.io/name=data-
extraction
traininghost  tm                      NodePort   10.108.140.158   <none>       32002:32002/TCP             4h1m   app.kubernetes.io/instance=tm,app.kubernetes.io/name=tm
```

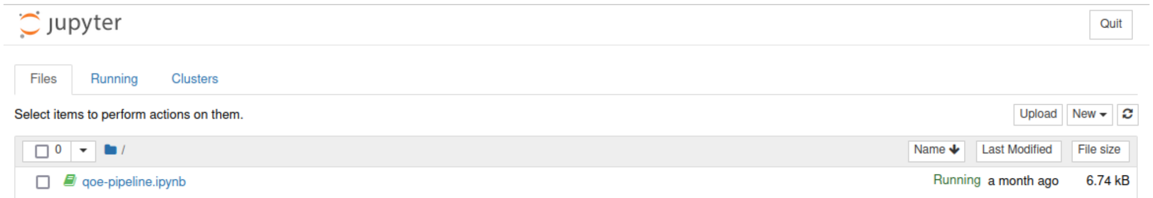
Port: 32088 to aiml-notebook



	Port	Local Address	Running Process	Origin
○	32002	localhost:32002		User Forwarded
○	32005	localhost:32005		User Forwarded
○	32088	localhost:32088		User Forwarded

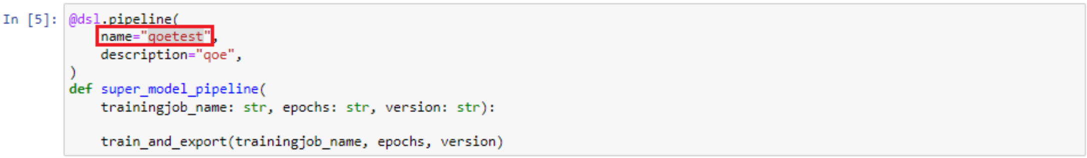
add 32088 port

- Port forward 32088 to aiml-notebook



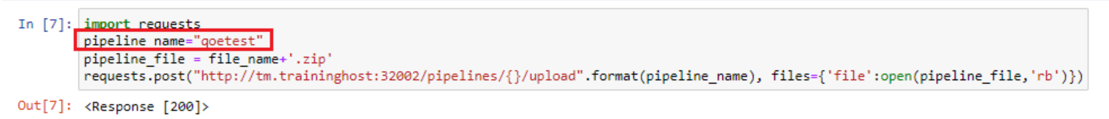
	Name		Last Modified	File size
□	qoe-pipeline.ipynb	Running	a month ago	6.74 kB

- After you click **“qoe-pipline.ipynb”**, you will see like this figure as the below.
  - **Step 1：** Modify **name** to the **“qoetest”**.



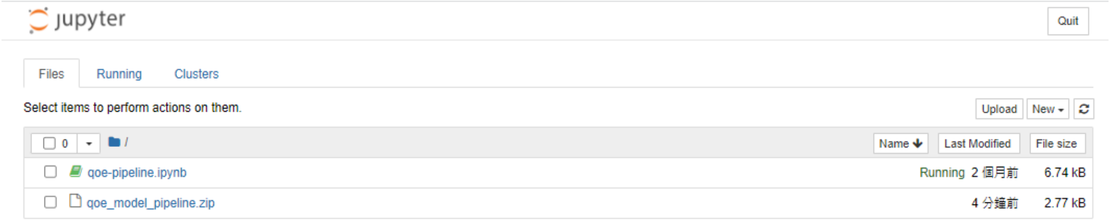
```
In [5]: @dsl.pipeline(
            name="qoetest",
            description="qoe",
        )
        def super_model_pipeline(
            trainingjob_name: str, epochs: str, version: str):

            train_and_export(trainingjob_name, epochs, version)
```

- **Step 2：** Modify **pipeline_name** to the **“qoetest”** before running. If you successful you will recieve 200 response.



```
In [7]: import requests
        pipeline_name="qoetest"
        pipeline_file = file_name+'.zip'
        requests.post("http://tm.traininghost:32002/pipelines/{}/upload".format(pipeline_name), files={'file':open(pipeline_file,'rb')})

Out[7]: <Response [200]>
```

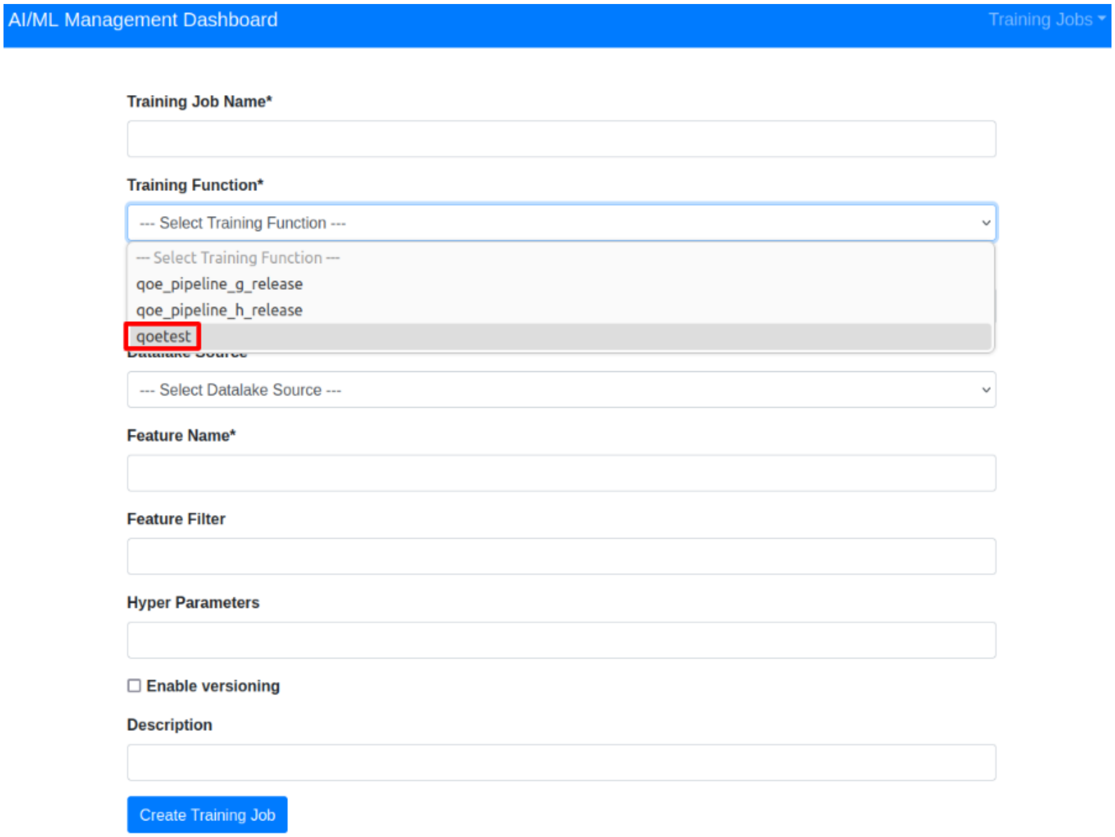
- **Step 3：** After you complete the above configuration, back off the previous page. You will see the **“qoe_model_pipeline.zip”** be created.



Select items to perform actions on them.

```
□ qoe-pipeline.ipynb        Running  2 個月前   6.74 kB
□ qoe_model_pipeline.zip             4 分鐘前   2.77 kB
```

- **Step 4：** Check the training function is correctly creat or not.



AI/ML Management Dashboard
Training Jobs
--- Select Training Function ---
--- Select Training Function ---
qoe_pipeline_g_release
qoe_pipeline_h_release
qoetest
--- Select Datalake Source ---
Create Training Job

Training Job Name*

Training Function*

Datalake Source

Feature Name*

Feature Filter

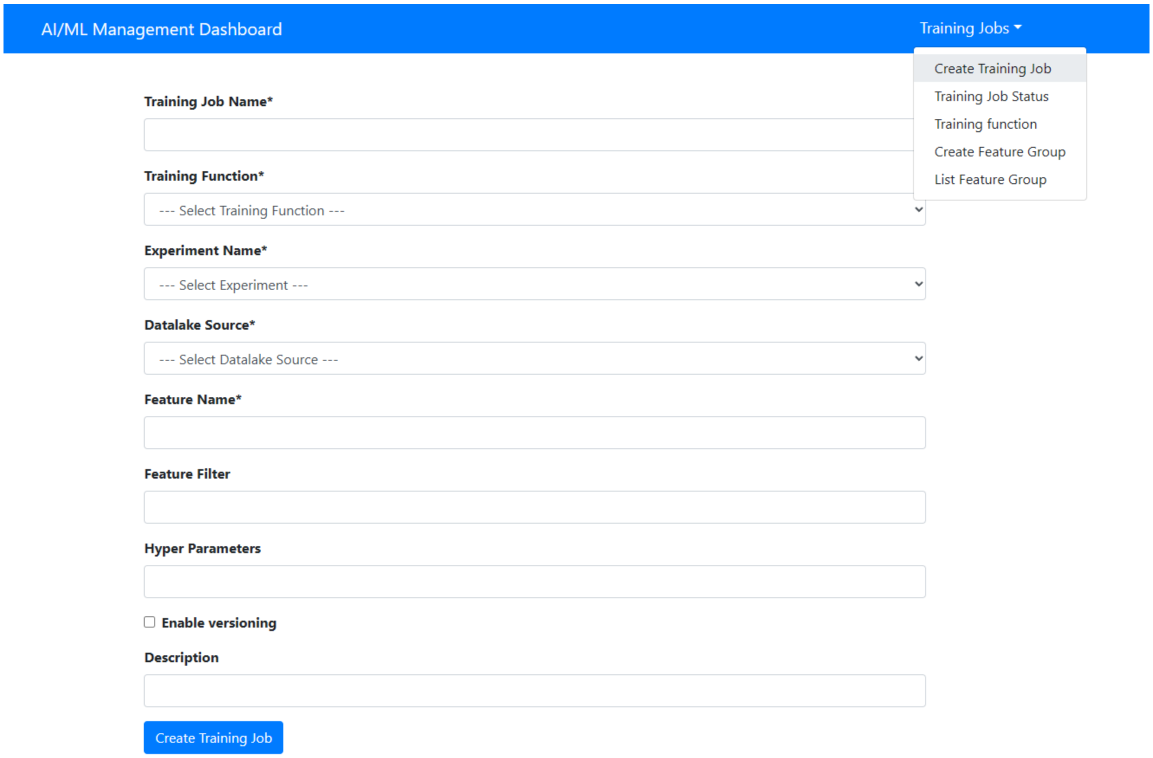
Hyper Parameters

□ Enable versioning

Description

▼ 3-2. Create training job

- Create an new training job on aiml-dashboard



Training Job Name*

Training Function*

--- Select Training Function ---

Experiment Name*

--- Select Experiment ---

Datalake Source*

--- Select Datalake Source ---

Feature Name*

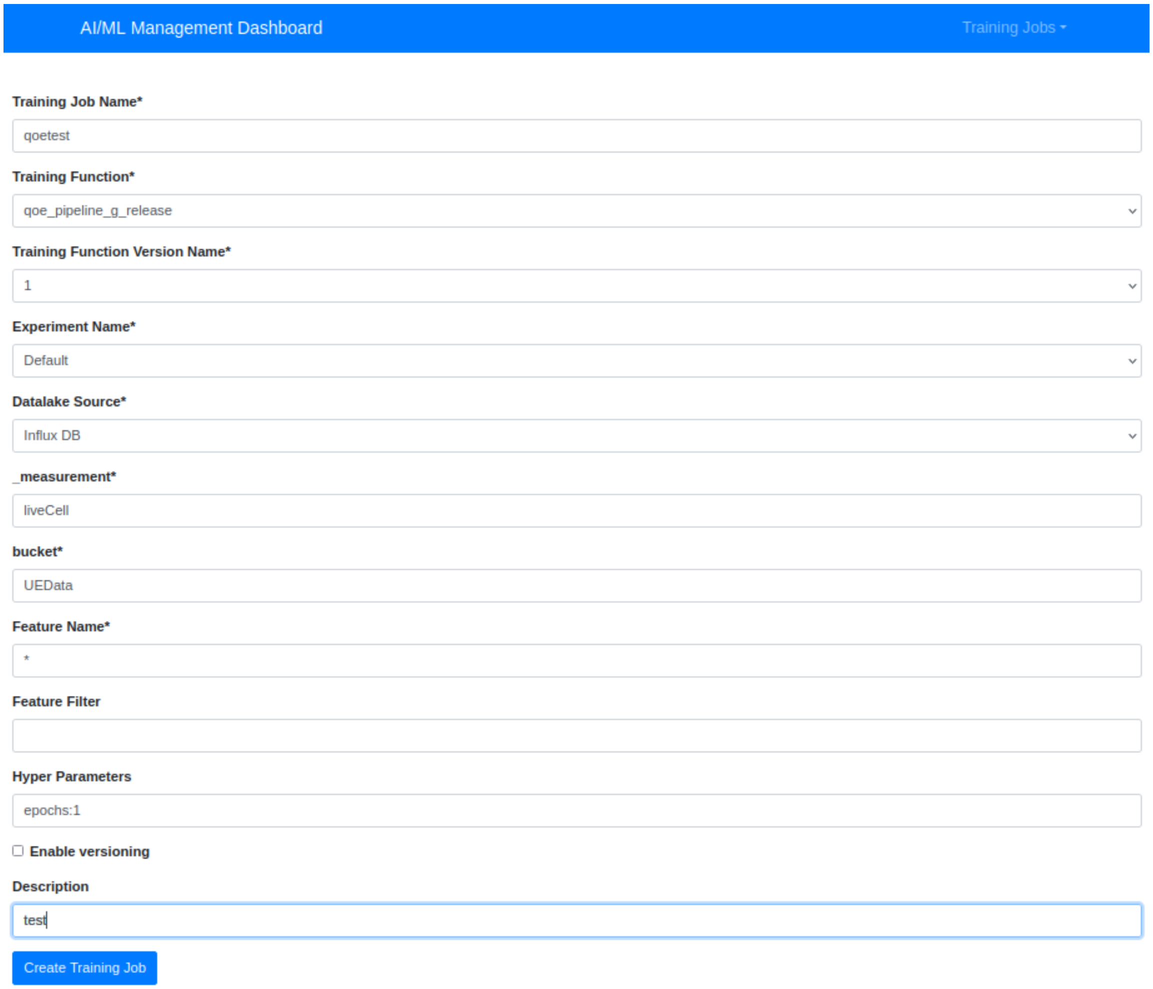
Feature Filter

Hyper Parameters

☐ Enable versioning

Description

- Use the default parameter by this figure. **“Training Functions”** which is that you previous create function.



Training Job Name*
qoetest

Training Function*
qoe_pipeline_g_release

Training Function Version Name*
1

Experiment Name*
Default

Datalake Source*
Influx DB

_measurement*
liveCell

bucket*
UEData

Feature Name*
*

Feature Filter

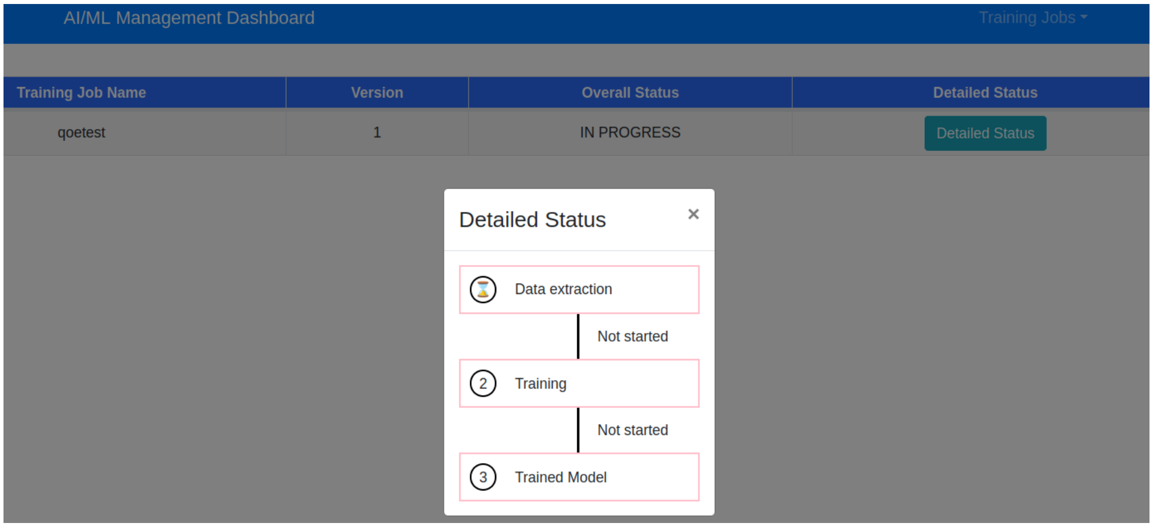
Hyper Parameters
epochs:1

□ Enable versioning

Description
test

Parameter	Value
Training Job Name	qoetest
Training Function	qoe_pipeline_h_release
Experiment Name	Default
Datalake Source	Influx DB
_measurement	test,ManagedElement=nodedntest,GNBDUFunction=1004,NRCellDU=c4_B2
bucket	pm-logg-bucket
Feature Name	*
Feature Filter	
Hyper Parameters	epochs:1
Description	test

- Back to the menu to select the **Detailed Status** to check model the training status



Training Job Name	Version	Overall Status	Detailed Status
qoetest	1	IN PROGRESS	Detailed Status

Detailed Status ×

⌛ Data extraction

Not started

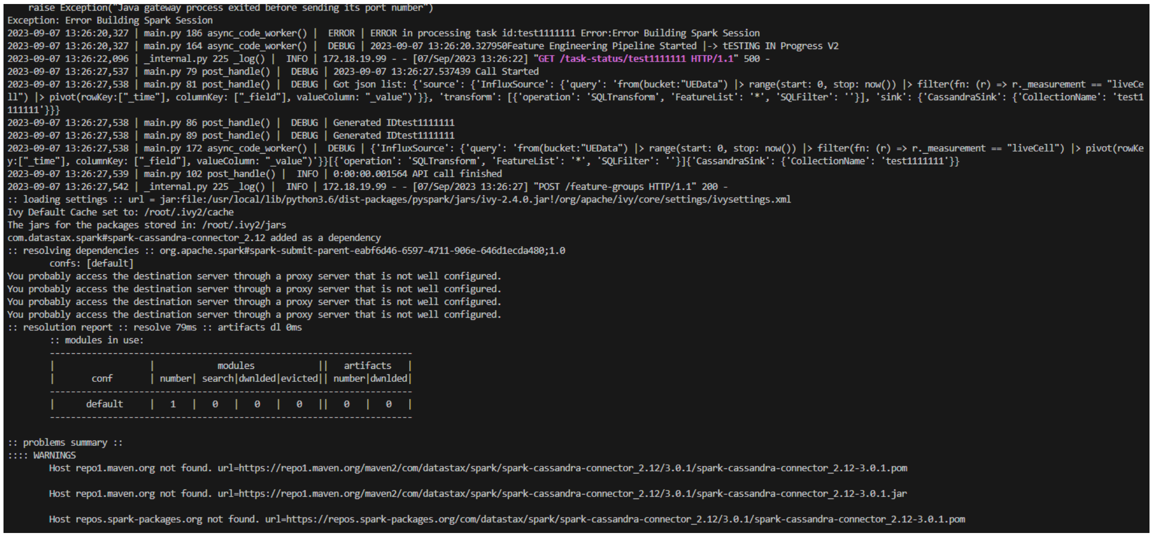
2 Training

Not started

3 Trained Model

▼ (Problem) The module cannot successfully downloaded in the data exaction pod.

- Data extraction pod error message (**CoreDNS Problem**)



```
    raise Exception("Java gateway process exited before sending its port number")
Exception: Error Building Spark Session
2023-09-07 13:26:20,327 | main.py 186 async_code_worker() |  ERROR | ERROR in processing task id:test1111111 Error:Error Building Spark Session
2023-09-07 13:26:20,327 | main.py 164 async_code_worker() |  DEBUG | 2023-09-07 13:26:20.327950Feature Engineering Pipeline Started |-> tESTING IN Progress V2
2023-09-07 13:26:22,096 | _internal.py 225 _log() |  INFO | 172.18.19.99 - - [07/Sep/2023 13:26:22] "GET /task-status/test1111111 HTTP/1.1" 500 -
2023-09-07 13:26:27,537 | main.py 79 post_handle() |  DEBUG | 2023-09-07 13:26:27.537439 Call Started
2023-09-07 13:26:27,538 | main.py 81 post_handle() |  DEBUG | Got json list: {'source': {'InfluxSource': {'query': 'from(bucket:"UEData") |> range(start: 0, stop: now()) |> filter(fn: (r) => r._measurement == "liveCe
ll") |> pivot(rowKey:["_time"], columnKey: ["_field"], valueColumn: "_value")'}}, 'transform': [{'operation': 'SQLTransform', 'FeatureList': '*', 'SQLFilter': ''}], 'sink': {'CassandraSink': {'CollectionName': 'test1
111111'}}}
2023-09-07 13:26:27,538 | main.py 86 post_handle() |  DEBUG | Generated IDtest1111111
2023-09-07 13:26:27,538 | main.py 89 post_handle() |  DEBUG | Generated IDtest1111111
2023-09-07 13:26:27,538 | main.py 172 async_code_worker() |  DEBUG | {'InfluxSource': {'query': 'from(bucket:"UEData") |> range(start: 0, stop: now()) |> filter(fn: (r) => r._measurement == "liveCell") |> pivot(rowKe
y:["_time"], columnKey: ["_field"], valueColumn: "_value")'}}[{'operation': 'SQLTransform', 'FeatureList': '*', 'SQLFilter': ''}]{'CassandraSink': {'CollectionName': 'test1111111'}}
2023-09-07 13:26:27,539 | main.py 102 post_handle() |  INFO | 0:00:00.001564 API call finished
2023-09-07 13:26:27,542 | _internal.py 225 _log() |  INFO | 172.18.19.99 - - [07/Sep/2023 13:26:27] "POST /feature-groups HTTP/1.1" 200 -
:: loading settings :: url = jar:file:/usr/local/lib/python3.6/dist-packages/pyspark/jars/ivy-2.4.0.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /root/.ivy2/cache
The jars for the packages stored in: /root/.ivy2/jars
com.datastax.spark#spark-cassandra-connector_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-eabf6d46-6597-4711-906e-646d1ecda480;1.0
        confs: [default]
You probably access the destination server through a proxy server that is not well configured.
You probably access the destination server through a proxy server that is not well configured.
You probably access the destination server through a proxy server that is not well configured.
You probably access the destination server through a proxy server that is not well configured.
:: resolution report :: resolve 79ms :: artifacts dl 0ms
        :: modules in use:
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   1   |   0   |   0   |   0   ||   0   |   0   |
        ---------------------------------------------------------------------

:: problems summary ::
:::: WARNINGS
        Host repo1.maven.org not found. url=https://repo1.maven.org/maven2/com/datastax/spark/spark-cassandra-connector_2.12/3.0.1/spark-cassandra-connector_2.12-3.0.1.pom

        Host repo1.maven.org not found. url=https://repo1.maven.org/maven2/com/datastax/spark/spark-cassandra-connector_2.12/3.0.1/spark-cassandra-connector_2.12-3.0.1.jar

        Host repos.spark-packages.org not found. url=https://repos.spark-packages.org/com/datastax/spark/spark-cassandra-connector_2.12/3.0.1/spark-cassandra-connector_2.12-3.0.1.pom
```

- To reslove **CoreDNS Problem** in kubernetes:
  - Step 1. Enter the data extraction podand **add nameserver 8.8.8.8**(Google's DNS server) to /etc/resolv.conf in the pod ,restart the data extraction pod and restart the training job again to download the essential module.

```
kubectl exec -it --namespace=traininghost data-extraction-755bcc4b8-drtdn -- bash
```

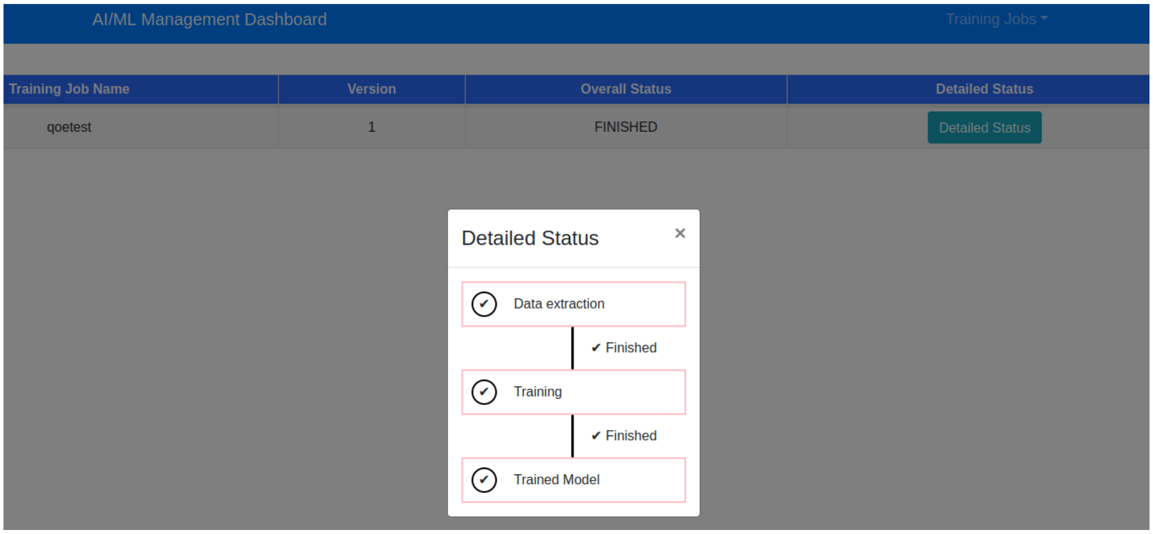
```
cat << EOF > /etc/resolv.conf
nameserver 8.8.8.8
nameserver 10.96.0.10
search traininghost.svc.cluster.local svc.cluster.local cluster.local localdomain
options ndots:5
EOF
```

```
kubectl rollout restart deployment data-extraction -n traininghost
```

  - Step 2. After the pod successfully downloads the module, enter the data extraction pod and **restore /etc/resolv.conf**.

```
cat << EOF > /etc/resolv.conf
nameserver 10.96.0.10
search traininghost.svc.cluster.local svc.cluster.local cluster.local localdomain
options ndots:5
EOF
```

- Re-execute the training job, wait for minutes then the model is complete.



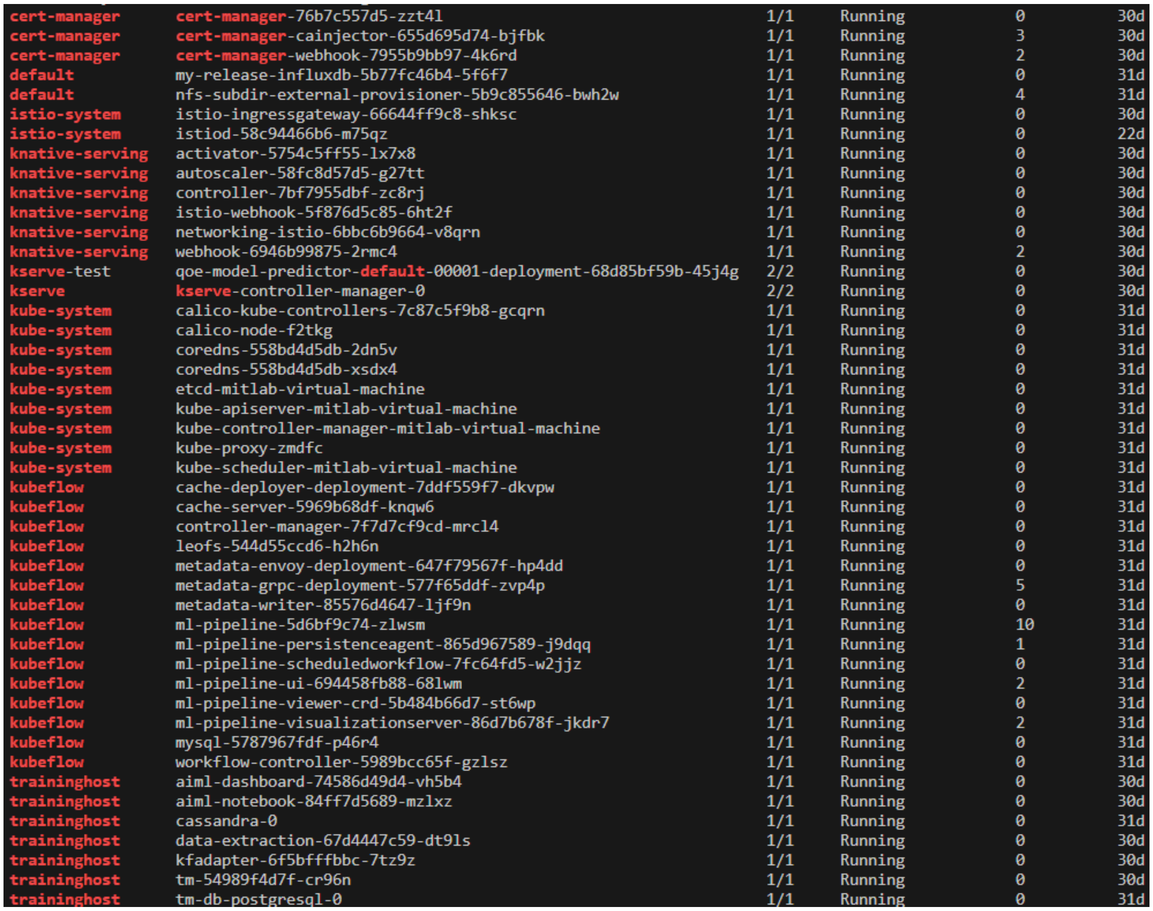
Training Job Name	Version	Overall Status	Detailed Status
qoetest	1	FINISHED	Detailed Status

3-3. Deploy trained qoe prediction model on KServe

- To install Kserve run the below commands.

```
./bin/install_kserve.sh
```

If you success you will see like this figure.



```
cert-manager      cert-manager-76b7c557d5-zzt4l                                   1/1     Running           0          30d
cert-manager      cert-manager-cainjector-655d695d74-bjfbk                        1/1     Running           3          30d
cert-manager      cert-manager-webhook-7955b9bb97-4k6rd                           1/1     Running           2          30d
default           my-release-influxdb-5b77fc46b4-5f6f7                            1/1     Running           0          31d
default           nfs-subdir-external-provisioner-5b9c855646-bwh2w                1/1     Running           4          31d
istio-system      istio-ingressgateway-66644ff9c8-shksc                           1/1     Running           0          30d
istio-system      istiod-58c94466b6-m75qz                                         1/1     Running           0          22d
knative-serving   activator-5754c5ff55-lx7x8                                      1/1     Running           0          30d
knative-serving   autoscaler-58fc8d57d5-g27tt                                     1/1     Running           0          30d
knative-serving   controller-7bf7955dbf-zc8rj                                     1/1     Running           0          30d
knative-serving   istio-webhook-5f876d5c85-6ht2f                                  1/1     Running           0          30d
knative-serving   networking-istio-6bbc6b9664-v8qrn                               1/1     Running           0          30d
knative-serving   webhook-6946b99875-2rmc4                                        1/1     Running           2          30d
kserve-test       qoe-model-predictor-default-00001-deployment-68d85bf59b-45j4g   2/2     Running           0          30d
kserve            kserve-controller-manager-0                                     2/2     Running           0          30d
kube-system       calico-kube-controllers-7c87c5f9b8-gcqrn                        1/1     Running           0          31d
kube-system       calico-node-f2tkg                                               1/1     Running           0          31d
kube-system       coredns-558bd4d5db-2dn5v                                        1/1     Running           0          31d
kube-system       coredns-558bd4d5db-xsdx4                                        1/1     Running           0          31d
kube-system       etcd-mitlab-virtual-machine                                     1/1     Running           0          31d
kube-system       kube-apiserver-mitlab-virtual-machine                           1/1     Running           0          31d
kube-system       kube-controller-manager-mitlab-virtual-machine                  1/1     Running           0          31d
kube-system       kube-proxy-zmdfc                                                1/1     Running           0          31d
kube-system       kube-scheduler-mitlab-virtual-machine                           1/1     Running           0          31d
kubeflow          cache-deployer-deployment-7ddf559f7-dkvpw                       1/1     Running           0          31d
kubeflow          cache-server-5969b68df-knqw6                                    1/1     Running           0          31d
kubeflow          controller-manager-7f7d7cf9cd-mrcl4                             1/1     Running           0          31d
kubeflow          leofs-544d55ccd6-h2h6n                                          1/1     Running           0          31d
kubeflow          metadata-envoy-deployment-647f79567f-hp4dd                      1/1     Running           0          31d
kubeflow          metadata-grpc-deployment-577f65ddf-zvp4p                        1/1     Running           5          31d
kubeflow          metadata-writer-85576d4647-ljf9n                                1/1     Running           0          31d
kubeflow          ml-pipeline-5d6bf9c74-zlwsm                                     1/1     Running           10         31d
kubeflow          ml-pipeline-persistenceagent-865d967589-j9dqq                   1/1     Running           1          31d
kubeflow          ml-pipeline-scheduledworkflow-7fc64fd5-w2jjz                    1/1     Running           0          31d
kubeflow          ml-pipeline-ui-694458fb88-68lwm                                 1/1     Running           2          31d
kubeflow          ml-pipeline-viewer-crd-5b484b66d7-st6wp                         1/1     Running           0          31d
kubeflow          ml-pipeline-visualizationserver-86d7b678f-jkdr7                 1/1     Running           2          31d
kubeflow          mysql-5787967fdf-p46r4                                          1/1     Running           0          31d
kubeflow          workflow-controller-5989bcc65f-gzlsz                            1/1     Running           0          31d
traininghost      aiml-dashboard-74586d49d4-vh5b4                                 1/1     Running           0          30d
traininghost      aiml-notebook-84ff7d5689-mzlxz                                  1/1     Running           0          30d
traininghost      cassandra-0                                                     1/1     Running           0          31d
traininghost      data-extraction-67d4447c59-dt9ls                                1/1     Running           0          30d
traininghost      kfadapter-6f5bfffbbc-7tz9z                                      1/1     Running           0          30d
traininghost      tm-54989f4d7f-cr96n                                             1/1     Running           0          30d
traininghost      tm-db-postgresql-0                                              1/1     Running           0          31d
```

- Create namespace using command below.

```
kubectl create namespace kserve-test
```

- Create qoe.yaml file with below contents.

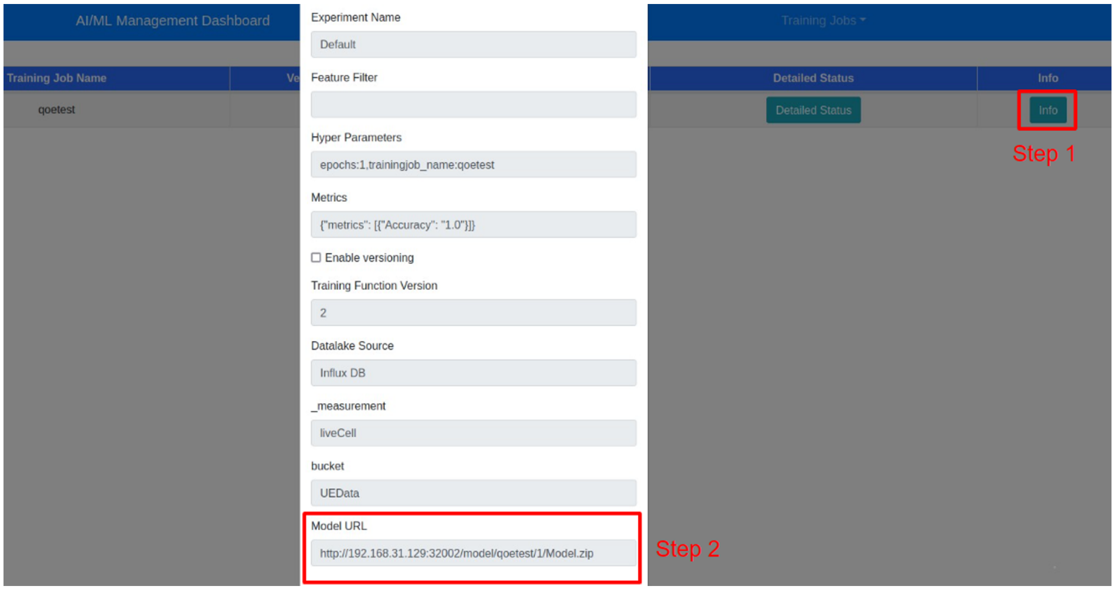
```
nano qoe.yaml
```

- Update the file like this figure.

```
apiVersion: "serving.kserve.io/v1beta1"
kind: "InferenceService"
metadata:
  name: qoe-model
spec:
  predictor:
    tensorflow:
      storageUri: "<update Model URL here>"
      runtimeVersion: "2.5.1"
      resources:
        requests:
          cpu: 0.1
          memory: 0.5Gi
        limits:
```

```
        cpu: 0.1
        memory: 0.5Gi
```

- Use the below step to get the model storage url.
  - Step 1. Click info.
  - Step 2. Copy the Model URL(storageUri).



qoetest

Experiment Name
Default

Feature Filter

Hyper Parameters
epochs:1,trainingjob_name:qoetest

Metrics
{"metrics": [{"Accuracy": "1.0"}]}

☐ Enable versioning

Training Function Version
2

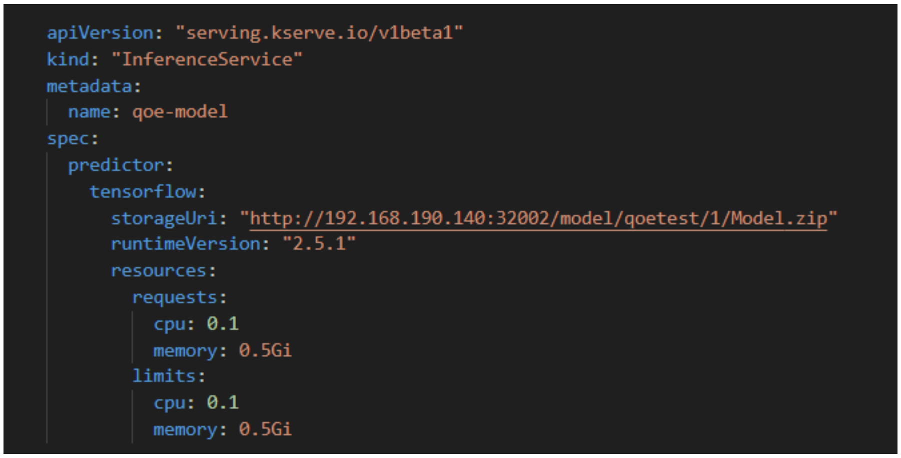
Datalake Source
Influx DB

_measurement
liveCell

bucket
UEData

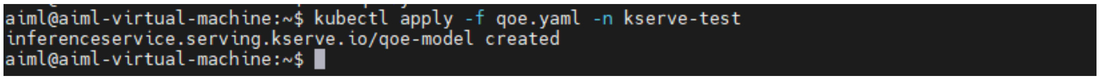
Model URL
http://192.168.31.129:32002/model/qoetest/1/Model.zip

  - Step 3. Update “storageUri” in qoe.yaml file.



```
apiVersion: "serving.kserve.io/v1beta1"
kind: "InferenceService"
metadata:
  name: qoe-model
spec:
  predictor:
    tensorflow:
      storageUri: "http://192.168.190.140:32002/model/qoetest/1/Model.zip"
      runtimeVersion: "2.5.1"
      resources:
        requests:
          cpu: 0.1
          memory: 0.5Gi
        limits:
          cpu: 0.1
          memory: 0.5Gi
```

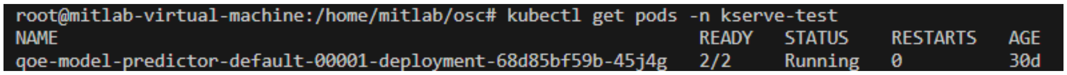
- To deploy model updated the Model URL in the qoe.yaml file and execute below command to deploy model.



```
aiml@aiml-virtual-machine:~$ kubectl apply -f qoe.yaml -n kserve-test
inferenceservice.serving.kserve.io/qoe-model created
aiml@aiml-virtual-machine:~$ 
```

- Check running state of pod using below command

```
kubectl get pods -n kserve-test
```

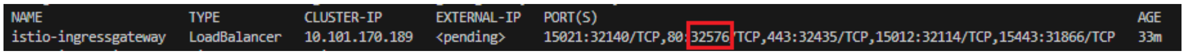


```
root@mitlab-virtual-machine:/home/mitlab/osc# kubectl get pods -n kserve-test
NAME                                                   READY   STATUS    RESTARTS   AGE
qoe-model-predictor-default-00001-deployment-68d85bf59b-45j4g   2/2     Running   0          30d
```

▼ Step 4. Test predictions using model deployed on Kserve

- Use below command to obtain Ingress port for Kserve.

```
kubectl get svc istio-ingressgateway -n istio-system
```



```
NAME                   TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)                                                                      AGE
istio-ingressgateway   LoadBalancer   10.101.170.189   <pending>     15021:32140/TCP,80:32576/TCP,443:32435/TCP,15012:32114/TCP,15443:31866/TCP   33m
```

- Create `predict.sh` file with following contents

```
nano predict.sh
```

- Copy the below content and update the **“IP of host”** where Kserve is deployed and ingress **“port”** of Kserve obtained using above method.

```
model_name=qoe-model
curl -v -H "Host: $model_name.kserve-test.example.com" http://"IP of where Kserve is deployed":"ingress port for Kserve"/v1/models/
```

For example:

```
model_name=qoe-model
curl -v -H "Host: $model_name.kserve-test.example.com" http://192.168.190.140":32576/v1/models/$model_name:predict -d @./input_qoe.
```

- After complete update, create sample data for predictions in file **input_qoe.json**. Add the following content in input_qoe.json file.

```
nano input_qoe.json
```

Add the following content in `input_qoe.json` file.

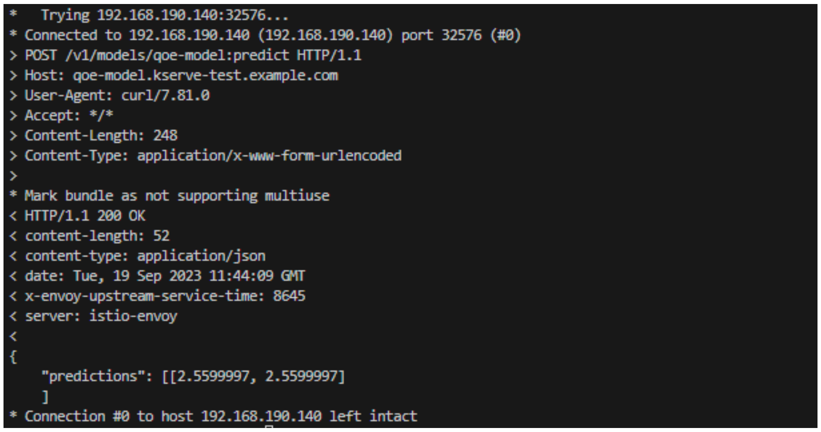
```
{"signature_name": "serving_default", "instances": [[[2.56, 2.56],
       [2.56, 2.56],
       [2.56, 2.56],
       [2.56, 2.56],
       [2.56, 2.56],
       [2.56, 2.56],
       [2.56, 2.56],
       [2.56, 2.56],
       [2.56, 2.56],
       [2.56, 2.56]]]}
```

- Use command below to trigger predictions.

```
source predict.sh
```

- **SUCCESSFUL RESULT**

  If you appear this information, you will see like below and that mean you complete the AI/ML Install.



```
*   Trying 192.168.190.140:32576...
* Connected to 192.168.190.140 (192.168.190.140) port 32576 (#0)
> POST /v1/models/qoe-model:predict HTTP/1.1
> Host: qoe-model.kserve-test.example.com
> User-Agent: curl/7.81.0
> Accept: */*
> Content-Length: 248
> Content-Type: application/x-www-form-urlencoded
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< content-length: 52
< content-type: application/json
< date: Tue, 19 Sep 2023 11:44:09 GMT
< x-envoy-upstream-service-time: 8645
< server: istio-envoy
<
{
    "predictions": [[2.5599997, 2.5599997]
    ]
* Connection #0 to host 192.168.190.140 left intact
```

▼ Step 5.Prepare Non-RT RIC DME as data source for AIMLFW

▼ 5-1. RANPM setup

- Download "nonrtric_plt_ranpm"

```
git clone "https://gerrit.o-ran-sc.org/r/nonrtric/plt/ranpm" && (cd "ranpm" && mkdir -p `git rev-parse --git-dir`/hooks/ && curl
```

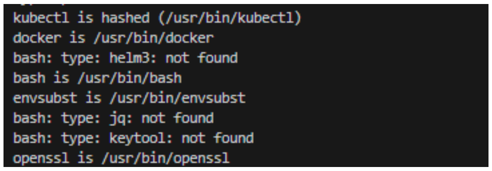
- Bring up the RANPM setup by following the steps mentioned in the file install/README.md present in the repository RANPM repository

> Requirements: helm3、bash、envsubst、jq、keytool、openssl

To check the requirement is installed or not

```
type kubectl
type docker
helm version
type bash
type envsubst
type jq
type keytool
type openssl
```

It appears that some of the required tools are not found (`helm3`, `jq`, `keytool`).



```
kubectl is hashed (/usr/bin/kubectl)
docker is /usr/bin/docker
bash: type: helm3: not found
bash is /usr/bin/bash
envsubst is /usr/bin/envsubst
bash: type: jq: not found
bash: type: keytool: not found
openssl is /usr/bin/openssl
```

  - **Install Helm 3**

    ```
    curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null
    sudo apt-get install apt-transport-https --yes
    echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian
    sudo apt-get update
    sudo apt-get install helm
    ```

  - **Install jq**

    ```
    sudo apt install jq
    ```

  - **Install keytool**

    ```
    sudo apt install openjdk-11-jdk  # Install Java 11
    ```

  - **Set JAVA_HOME** (Optional):

    ```
    export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64  # Adjust the path as needed
    export PATH=$PATH:$JAVA_HOME/bin
    ```

- To check the Helm version

  ```
  helm version
  type jq
  type keytool
  ```

- Build the following images

  1. ranpm/https-server

     - Build for docker or local kubernetes

       ```
       cd /home/mitlab/osc/aimlfw-dep/ranpm/https-server
       ./build.sh no-push
       ```

     - Output information

       ```
       Digest: sha256:73c225bc5e2353f20dbe0466819b70a51a114a93bfe4af035a3bb9e1ecdd4107
       ...
       Successfully built 0c36df07ed87
       Successfully tagged pm-https-server:latest
       BUILD OK
       IMAGE OK: pm-https-server:latest
       DON
       ```

       Successfully built 0c36df07ed87
       Successfully tagged pm-https-server:latest
       BUILD OK
       IMAGE OK: pm-https-server:latest
       DONE

  2. pm-rapp

- Build for local

```
cd /home/mitlab/osc/aimlfw-dep/ranpm/pm-rapp
./build.sh no-push
```

- Output information

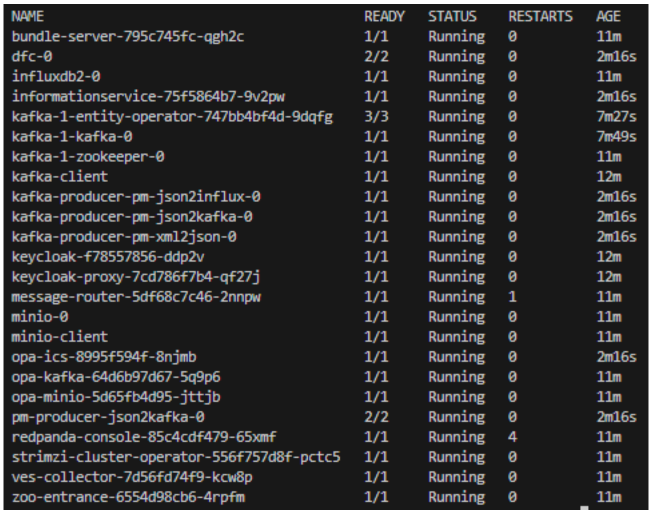
```
Digest: sha256:46c5b9bd3e3efff512e28350766b54355fce6337a0b44ba3f822ab918eca4520
Status: Downloaded newer image for gcr.io/distroless/base-debian11:latest
...
Successfully built a36daf1962c2
Successfully tagged pm-rapp:latest
BUILD OK
IMAGE OK: pm-rapp:latest
DONE
```

- **Installation**
  - Install install-nrt.sh : Installs the main parts of the ranpm setup

```
cd /home/mitlab/osc/aimlfw-dep/ranpm/install
./install-nrt.sh
```

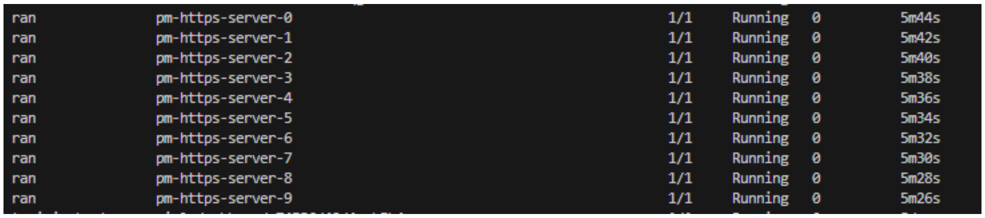
  - Verify that all pods are in status Running

```
kubectl get po -n nonrtric
```



```
NAME                                        READY   STATUS    RESTARTS   AGE
bundle-server-795c745fc-qgh2c               1/1     Running   0          11m
dfc-0                                       2/2     Running   0          2m16s
influxdb2-0                                 1/1     Running   0          11m
informationservice-75f5864b7-9v2pw          1/1     Running   0          2m16s
kafka-1-entity-operator-747bb4bf4d-9dqfg    3/3     Running   0          7m27s
kafka-1-kafka-0                             1/1     Running   0          7m49s
kafka-1-zookeeper-0                         1/1     Running   0          11m
kafka-client                                1/1     Running   0          12m
kafka-producer-pm-json2influx-0             1/1     Running   0          2m16s
kafka-producer-pm-json2kafka-0              1/1     Running   0          2m16s
kafka-producer-pm-xml2json-0                1/1     Running   0          2m16s
keycloak-f78557856-ddp2v                    1/1     Running   0          12m
keycloak-proxy-7cd786f7b4-qf27j             1/1     Running   0          12m
message-router-5df68c7c46-2nnpw             1/1     Running   1          11m
minio-0                                     1/1     Running   0          11m
minio-client                                1/1     Running   0          11m
opa-ics-8995f594f-8njmb                     1/1     Running   0          2m16s
opa-kafka-64d6b97d67-5q9p6                  1/1     Running   0          11m
opa-minio-5d65fb4d95-jttjb                  1/1     Running   0          11m
pm-producer-json2kafka-0                    2/2     Running   0          2m16s
redpanda-console-85c4cdf479-65xmf           1/1     Running   4          11m
strimzi-cluster-operator-556f757d8f-pctc5   1/1     Running   0          11m
ves-collector-7d56fd74f9-kcw8p              1/1     Running   0          11m
zoo-entrance-6554d98cb6-4rpfm               1/1     Running   0          11m
```

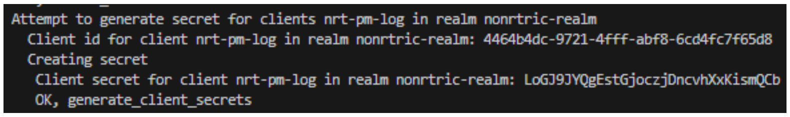
```
kubectl get po -n ran
```



```
ran            pm-https-server-0                                   1/1     Running   0          5m44s
ran            pm-https-server-1                                   1/1     Running   0          5m42s
ran            pm-https-server-2                                   1/1     Running   0          5m40s
ran            pm-https-server-3                                   1/1     Running   0          5m38s
ran            pm-https-server-4                                   1/1     Running   0          5m36s
ran            pm-https-server-5                                   1/1     Running   0          5m34s
ran            pm-https-server-6                                   1/1     Running   0          5m32s
ran            pm-https-server-7                                   1/1     Running   0          5m30s
ran            pm-https-server-8                                   1/1     Running   0          5m28s
ran            pm-https-server-9                                   1/1     Running   0          5m26s
```

- Install install-pm-log.sh : Installs the producer for influx db

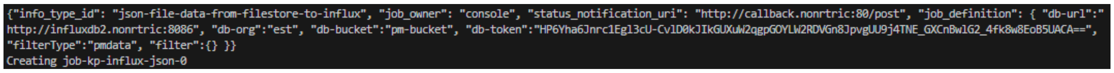
  ```
  ./install-pm-log.sh
  ```

  ```
  Attempt to generate secret for clients nrt-pm-log in realm nonrtric-realm
    Client id for client nrt-pm-log in realm nonrtric-realm: 4464b4dc-9721-4fff-abf8-6cd4fc7f65d8
    Creating secret
     Client secret for client nrt-pm-log in realm nonrtric-realm: LoGJ9JYQgEstGjoczjDncvhXxKismQCb
     OK, generate_client_secrets
  ```

- Install install-pm-influx-job.sh : Sets up an alternative job to produce data stored in influx db

  ```
  ./install-pm-influx-job.sh
  ```

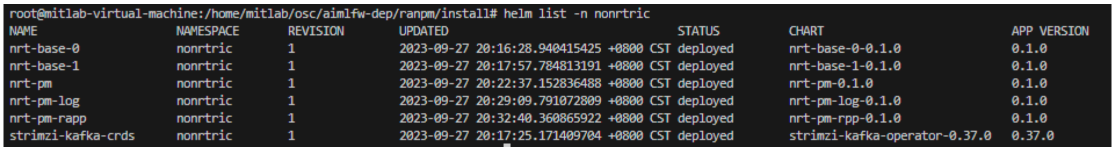
  ```
  {"info_type_id": "json-file-data-from-filestore-to-influx", "job_owner": "console", "status_notification_uri": "http://callback.nonrtric:80/post", "job_definition": { "db-url":"
  http://influxdb2.nonrtric:8086", "db-org":"est", "db-bucket":"pm-bucket", "db-token":"HP6Yha6Jnrc1Egl3cU-CvlD0kJIkGUXuW2qgpGOYLW2RDVGn8JpvgUU9j4TNE_GXCnBwlG2_4fk8w8EoB5UACA==",
  "filterType":"pmdata", "filter":{} }}
  Creating job-kp-influx-json-0
  ```

- Install install-pm-rapp.sh : Installs a rapp that subscribe and print out received data

  ```
  ./install-pm-rapp.sh
  ```

- **Check the Status**

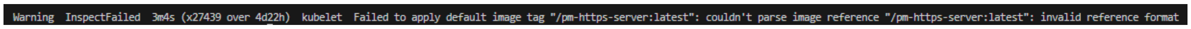
  ```
  helm list -n nonrtric
  ```

  ```
  root@mitlab-virtual-machine:/home/mitlab/osc/aimlfw-dep/ranpm/install# helm list -n nonrtric
  NAME                NAMESPACE       REVISION        UPDATED                                 STATUS          CHART                           APP VERSION
  nrt-base-0          nonrtric        1               2023-09-27 20:16:28.940415425 +0800 CST deployed        nrt-base-0-0.1.0                0.1.0
  nrt-base-1          nonrtric        1               2023-09-27 20:17:57.784813191 +0800 CST deployed        nrt-base-1-0.1.0                0.1.0
  nrt-pm              nonrtric        1               2023-09-27 20:22:37.152836488 +0800 CST deployed        nrt-pm-0.1.0                    0.1.0
  nrt-pm-log          nonrtric        1               2023-09-27 20:29:09.791072809 +0800 CST deployed        nrt-pm-log-0.1.0                0.1.0
  nrt-pm-rapp         nonrtric        1               2023-09-27 20:32:40.360865922 +0800 CST deployed        nrt-pm-rpp-0.1.0                0.1.0
  strimzi-kafka-crds  nonrtric        1               2023-09-27 20:17:25.171409704 +0800 CST deployed        strimzi-kafka-operator-0.37.0   0.37.0
  ```

▼ **(Problem)** Failed to apply default image tag

Problem: Failed to apply default image tag "/pm-https-server:latest": couldn't parse image reference "/pm-https-server:latest": invalid reference format



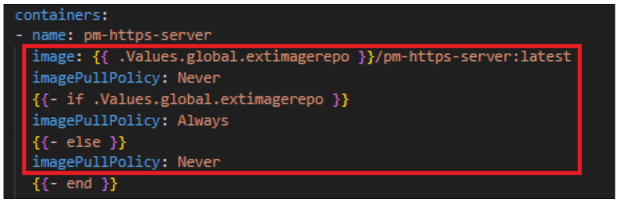
```
Warning  InspectFailed  3m4s (x27439 over 4d22h)  kubelet  Failed to apply default image tag "/pm-https-server:latest": couldn't parse image reference "/pm-https-server:latest": invalid reference format
```

A4.

- Discover `app-deployment.yaml` {{ .Values.global.extimagerepo }} that the **extimagerepo** value of `ranpm/install/helm/global-values.yaml` is null, so delete it.

  Resolve: Revise `ranpm/install/helm/ran/templates/app-deployment.yaml`

  Delete {{ .Values.global.extimagerepo }

  ```
  containers:
  - name: pm-https-server
    image: {{ .Values.global.extimagerepo }}/pm-https-server:latest
    imagePullPolicy: Never
    {{- if .Values.global.extimagerepo }}
    imagePullPolicy: Always
    {{- else }}
    imagePullPolicy: Never
    {{- end }}
  ```

  ```
  containers:
        - name: pm-https-server
  ```

```
        image: pm-https-server:latest
        imagePullPolicy: IfNotPresent
```

- In addition, pm-rapp has the same problem, so modify `ranpm/install/helm/nrt-pm-rapp/templates/app-pod.yaml` as well.

▼ 5-2. Create Feature Group

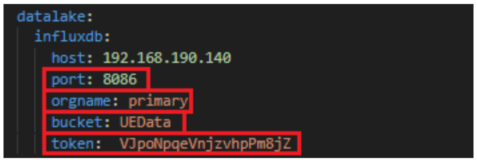
- Get Influx DB access token

```
cd aimlfw-dep/demos/hrelease/scripts
./get_access_tokens.sh
```

Influx DB token

```
UbTgwNGUkESZpdNNY4MQdl5kDnY7Al1MNlBjJ_j7SbYKp9rnQl-vAIWJbNSaWbqcoNGImtpLBJo7vMl-xii79Q==UbTgwNGUkESZpdNNY4MQdl5kDnY7Al1MNlBjJ_j7
```

- Update the RECIPE file (`RECIPE_EXAMPLE/example_recipe_latest_stable.yaml`)



```
datalake:
  influxdb:
    host: 192.168.190.140
    port: 8086
    orgname: primary
    bucket: UEData
    token:  VJpoNpqeVnjzvhpPm8jZ
```

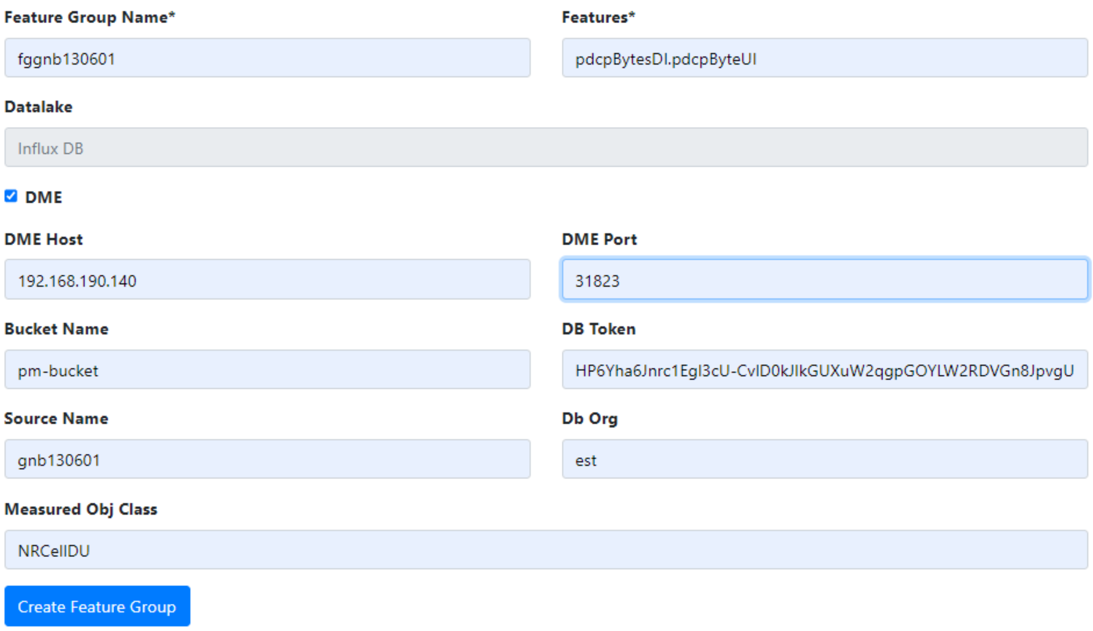
```
datalake:
  influxdb:
    host: 192.168.190.140
    port: 31812
    orgname: est
    bucket: pm-bucket
    token: UbTgwNGUkESZpdNNY4MQdl5kDnY7Al1MNlBjJ_j7SbYKp9rnQl-vAIWJbNSaWbqcoNGImtpLBJo7vMl-xii79Q==UbTgwNGUkESZpdNNY4MQdl5kDnY7A
```

```
bin/uninstall.sh
bin/install.sh -f RECIPE_EXAMPLE/example_recipe_latest_stable.yaml
```

```
cd /home/mitlab/osc/aimlfw-dep/demos/hrelease/scripts
./prepare_env_aimlfw_access.sh
```

Execute the below script

- Create Feature Group in AI/ML Management Dashboard



Feature Group Name*

fggnb130601

Features*

pdcpBytesDl.pdcpByteUl

Datalake

Influx DB

☑ **DME**

DME Host

192.168.190.140

DME Port

31823

Bucket Name

pm-bucket

DB Token

HP6Yha6Jnrc1Egl3cU-CvlD0kJIkGUXuW2qgpGOYLW2RDVGn8JpvgU

Source Name

gnb130601

Db Org

est

Measured Obj Class

NRCellDU

```
Feature Group Name: fggnb130601
Features: pdcpBytesDl, pdcpBytesUl
DME Port: 31823
Bucket Name: pm-bucket
Source Name: gnb130601
Db Org: est
Measured Obj Class: NRCellDU
```

▼ 5-3. Push QoE data

- Execute below script to push qoe data into ranpm setup

```
./push_qoe_data.sh  <source name mentioned when creating feature group> <Number of rows> <Cell Identity>
```

For example

```
./push_qoe_data.sh gnb130601 30 c4/B2
```

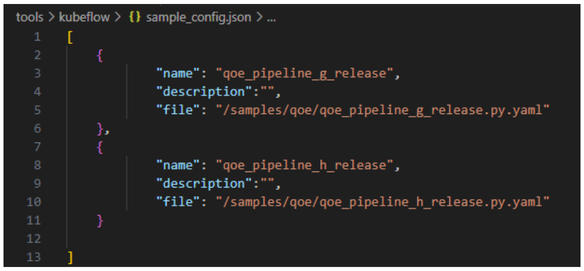
- Check if data is upload correctly

```
kubectl exec -it influxdb2-0 -n nonrtric -- bash
influx query 'from(bucket: "pm-bucket") |> range(start: -1000000000000000000d)' |grep pdcpBytesDl
```

Problem

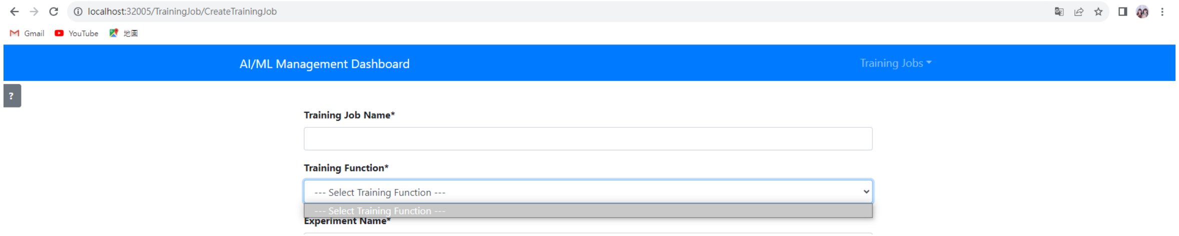
▼ Q1. When creating training job, Training Function is not pushed to AI/ML Management Dashboard

- Normally, Training Function must have qoe_pipeline_g_release and qoe_pipeline_h_release



```
[
    {
        "name": "qoe_pipeline_g_release",
        "description":"",
        "file": "/samples/qoe/qoe_pipeline_g_release.py.yaml"
    },
    {
        "name": "qoe_pipeline_h_release",
        "description":"",
        "file": "/samples/qoe/qoe_pipeline_h_release.py.yaml"
    }

]
```

- But Training Function is empty


localhost:32005/TrainingJob/CreateTrainingJob
Gmail
YouTube
地图
AI/ML Management Dashboard
Training Jobs ▾
?

Training Job Name*

Training Function*

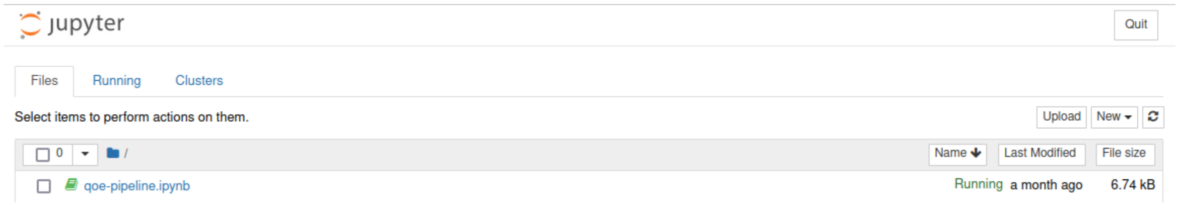
--- Select Training Function ---

--- Select Training Function ---

Experiment Name*

A1. After doing the following steps, you can successfully create Training Function.

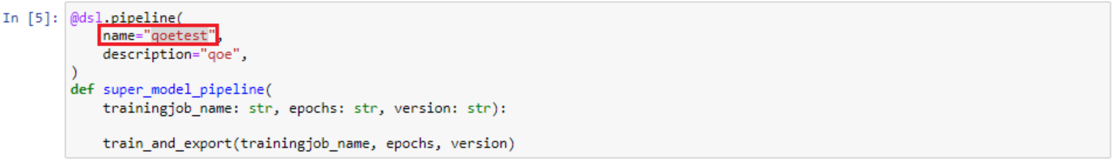
- Port forward 32088 to aiml-notebook



Select items to perform actions on them.

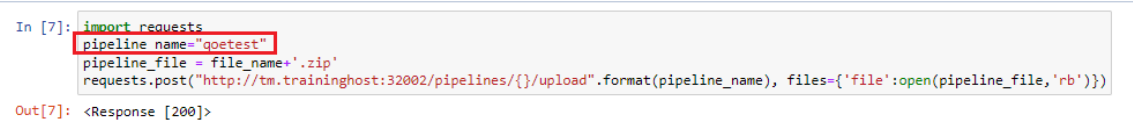
□ qoe-pipeline.ipynb	Running	a month ago	6.74 kB

- After you click **“qoe-pipline.ipynb”**, you will see like this figure as the below.
    - **Step 1：** Modify **name** to the **“qoetest”**.



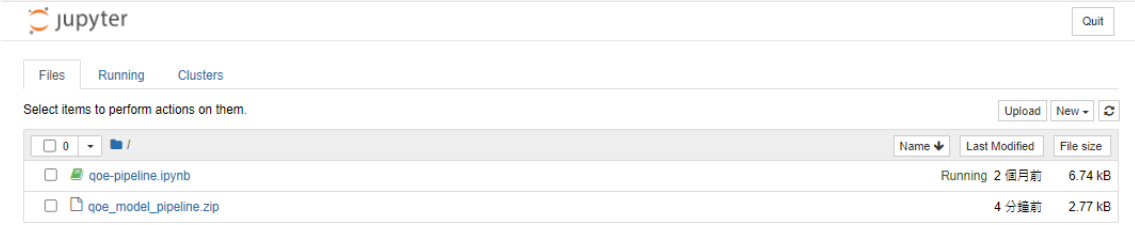
```
In [5]: @dsl.pipeline(
            name="qoetest",
            description="qoe",
        )
        def super_model_pipeline(
            trainingjob_name: str, epochs: str, version: str):

            train_and_export(trainingjob_name, epochs, version)
```

- **Step 2：** Modify **pipeline_name** to the **“qoetest”** before running. If you successful you will recieve 200 response.



```
In [7]: import requests
        pipeline_name="qoetest"
        pipeline_file = file_name+'.zip'
        requests.post("http://tm.traininghost:32002/pipelines/{}/upload".format(pipeline_name), files={'file':open(pipeline_file,'rb')})

Out[7]: <Response [200]>
```

- **Step 3：** After you complete the above configuration, back off the previous page. You will see the **“qoe_model_pipeline.zip”** be created.

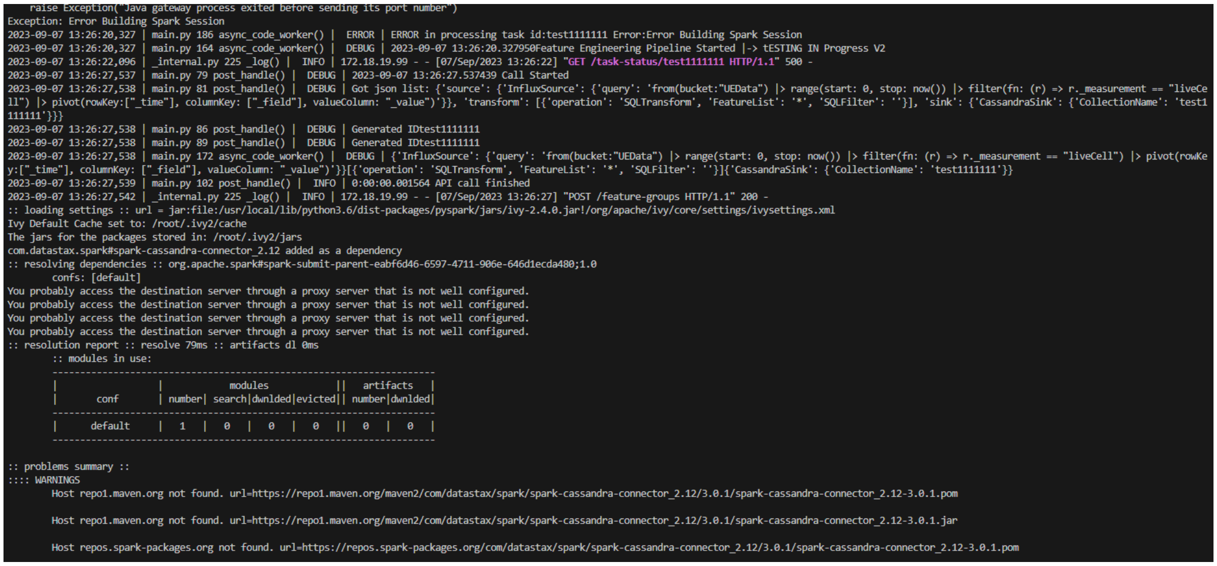


Select items to perform actions on them.

Name	Last Modified	File size
qoe-pipeline.ipynb	Running 2 個月前	6.74 kB
qoe_model_pipeline.zip	4 分鐘前	2.77 kB

- **Step 4：** Check the training function is correctly creat or not.

▼ Q2. Data extraction pod cannot download module (host resolving problem)



```
    raise Exception("Java gateway process exited before sending its port number")
Exception: Error Building Spark Session
2023-09-07 13:26:20,327 | main.py 186 async_code_worker() |  ERROR | ERROR in processing task id:test1111111 Error:Error Building Spark Session
2023-09-07 13:26:20,327 | main.py 164 async_code_worker() |  DEBUG | 2023-09-07 13:26:20.327950Feature Engineering Pipeline Started |-> tESTING IN Progress V2
2023-09-07 13:26:22,096 | _internal.py 225 _log() |  INFO | 172.18.19.99 - - [07/Sep/2023 13:26:22] "GET /task-status/test1111111 HTTP/1.1" 500 -
2023-09-07 13:26:27,537 | main.py 79 post_handle() |  DEBUG | 2023-09-07 13:26:27.537439 Call Started
2023-09-07 13:26:27,538 | main.py 81 post_handle() |  DEBUG | Got json list: {'source': {'InfluxSource': {'query': 'from(bucket:"UEData") |> range(start: 0, stop: now()) |> filter(fn: (r) => r._measurement == "liveCe
ll") |> pivot(rowKey:["_time"], columnKey: ["_field"], valueColumn: "_value")'}}, 'transform': [{'operation': 'SQLTransform', 'FeatureList': '*', 'SQLFilter': ''}], 'sink': {'CassandraSink': {'CollectionName': 'test1
111111'}}}
2023-09-07 13:26:27,538 | main.py 86 post_handle() |  DEBUG | Generated IDtest1111111
2023-09-07 13:26:27,538 | main.py 89 post_handle() |  DEBUG | Generated IDtest1111111
2023-09-07 13:26:27,538 | main.py 172 async_code_worker() |  DEBUG | {'InfluxSource': {'query': 'from(bucket:"UEData") |> range(start: 0, stop: now()) |> filter(fn: (r) => r._measurement == "liveCell") |> pivot(rowKe
y:["_time"], columnKey: ["_field"], valueColumn: "_value")'}}[{'operation': 'SQLTransform', 'FeatureList': '*', 'SQLFilter': ''}]{'CassandraSink': {'CollectionName': 'test1111111'}}
2023-09-07 13:26:27,539 | main.py 102 post_handle() |  INFO | 0:00:00.001564 API call finished
2023-09-07 13:26:27,542 | _internal.py 225 _log() |  INFO | 172.18.19.99 - - [07/Sep/2023 13:26:27] "POST /feature-groups HTTP/1.1" 200 -
:: loading settings :: url = jar:file:/usr/local/lib/python3.6/dist-packages/pyspark/jars/ivy-2.4.0.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /root/.ivy2/cache
The jars for the packages stored in: /root/.ivy2/jars
com.datastax.spark#spark-cassandra-connector_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-eabf6d46-6597-4711-906e-646d1ecda480;1.0
        confs: [default]
You probably access the destination server through a proxy server that is not well configured.
You probably access the destination server through a proxy server that is not well configured.
You probably access the destination server through a proxy server that is not well configured.
You probably access the destination server through a proxy server that is not well configured.
:: resolution report :: resolve 79ms :: artifacts dl 0ms
        :: modules in use:
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   1   |   0   |   0   |   0   ||   0   |   0   |
        ---------------------------------------------------------------------

:: problems summary ::
:::: WARNINGS
        Host repo1.maven.org not found. url=https://repo1.maven.org/maven2/com/datastax/spark/spark-cassandra-connector_2.12/3.0.1/spark-cassandra-connector_2.12-3.0.1.pom

        Host repo1.maven.org not found. url=https://repo1.maven.org/maven2/com/datastax/spark/spark-cassandra-connector_2.12/3.0.1/spark-cassandra-connector_2.12-3.0.1.jar

        Host repos.spark-packages.org not found. url=https://repos.spark-packages.org/com/datastax/spark/spark-cassandra-connector_2.12/3.0.1/spark-cassandra-connector_2.12-3.0.1.pom
```

A2.
For the coredns problem in the data extraction pod, add **nameserver 8.8.8.8** to /etc/resolv.conf in the pod and add Google's dns to the pod to download the module.

- To reslove **CoreDNS Problem** in kubernetes:
  - Step 1. Enter the data extraction podand **add nameserver 8.8.8.8**(Google's DNS server) to /etc/resolv.conf in the pod ,restart the data extraction pod and restart the training job again to download the essential module.

    ```
    kubectl exec -it --namespace=traininghost data-extraction-755bcc4b8-drtdn -- bash
    ```

    ```
    cat << EOF > /etc/resolv.conf
    nameserver 8.8.8.8
    nameserver 10.96.0.10
    search traininghost.svc.cluster.local svc.cluster.local cluster.local localdomain
    options ndots:5
    EOF
    ```

    ```
    kubectl rollout restart deployment data-extraction -n traininghost
    ```

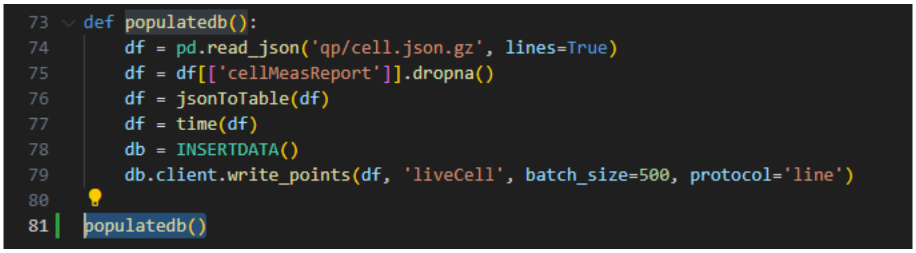
  - Step 2. After the pod successfully downloads the module, enter the data extraction pod and **restore /etc/resolv.conf**.

    ```
    cat << EOF > /etc/resolv.conf
    nameserver 10.96.0.10
    search traininghost.svc.cluster.local svc.cluster.local cluster.local localdomain
    ```

```
options ndots:5
EOF
```

▼ Q3. After inserting data into Influx DB, querying the Influx DB data did not find the data.

- A3-1.
  The latest version of `insert.py` seems to be missing the call to `populatedb()`. After manually adding the `populatedb()` call, InfluxDB started to populate with data.

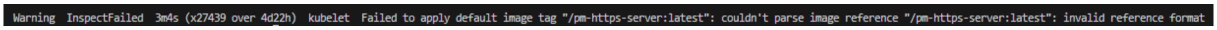


```
def populatedb():
    df = pd.read_json('qp/cell.json.gz', lines=True)
    df = df[['cellMeasReport']].dropna()
    df = jsonToTable(df)
    df = time(df)
    db = INSERTDATA()
    db.client.write_points(df, 'liveCell', batch_size=500, protocol='line')

populatedb()
```

- A3-2.
  After waiting for many hours, the data appeared.

▼ Q4. Failed to apply default image tag

Problem: Failed to apply default image tag "/pm-https-server:latest": couldn't parse image reference "/pm-https-server:latest": invalid reference format



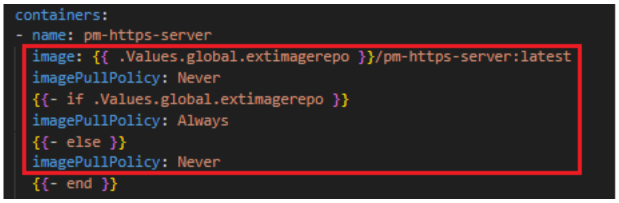
```
Warning  InspectFailed  3m4s (x27439 over 4d22h)  kubelet  Failed to apply default image tag "/pm-https-server:latest": couldn't parse image reference "/pm-https-server:latest": invalid reference format
```

A4.

- Discover `app-deployment.yaml` {{ .Values.global.extimagerepo }} that the **extimagerepo** value of `ranpm/install/helm/global-values.yaml` is null, so delete it.

  Resolve: Revise `ranpm/install/helm/ran/templates/app-deployment.yaml`

  Delete {{ .Values.global.extimagerepo }



```
containers:
- name: pm-https-server
  image: {{ .Values.global.extimagerepo }}/pm-https-server:latest
  imagePullPolicy: Never
  {{- if .Values.global.extimagerepo }}
  imagePullPolicy: Always
  {{- else }}
  imagePullPolicy: Never
  {{- end }}
```

```
containers:
      - name: pm-https-server
        image: pm-https-server:latest
        imagePullPolicy: IfNotPresent
```

- In addition, pm-rapp has the same problem, so modify `ranpm/install/helm/nrt-pm-rapp/templates/app-pod.yaml` as well.