AFS-docs Documentation

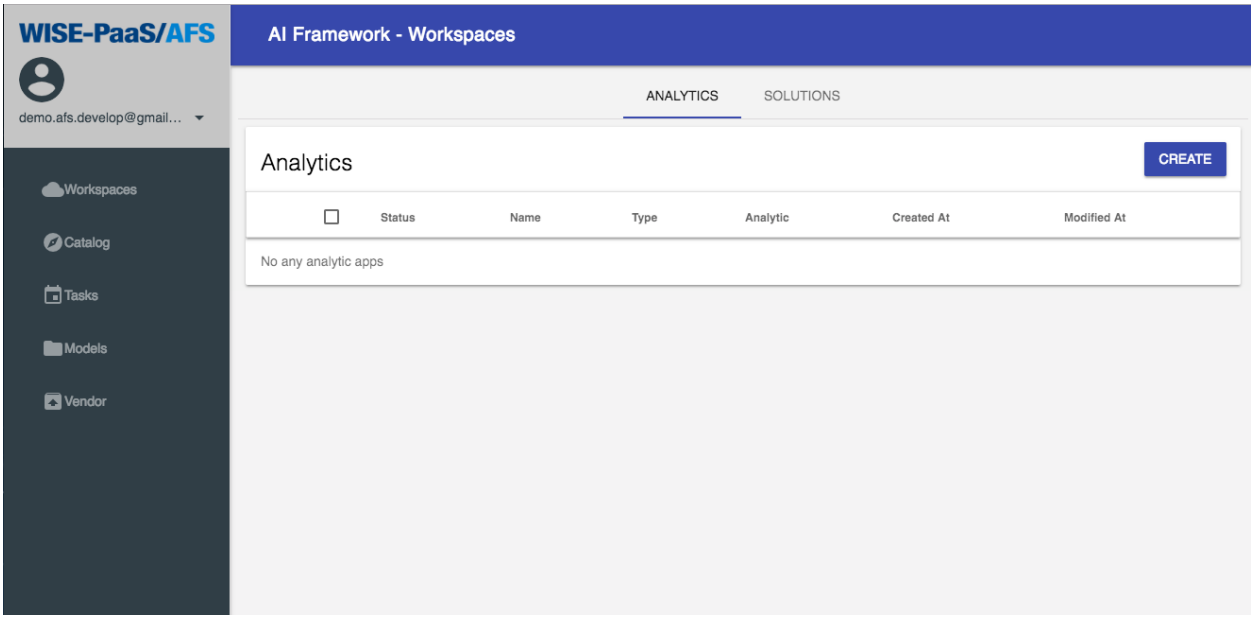
Jacobchen

Feb 20, 2019

Portal

CHAPTER 1

Workspace



1.1 Analytics

1.1.1 Online Code IDE

In **AFS**, we provide a powerful **Online Code IDE** based on Jupyter to develop your analytic on the cloud.

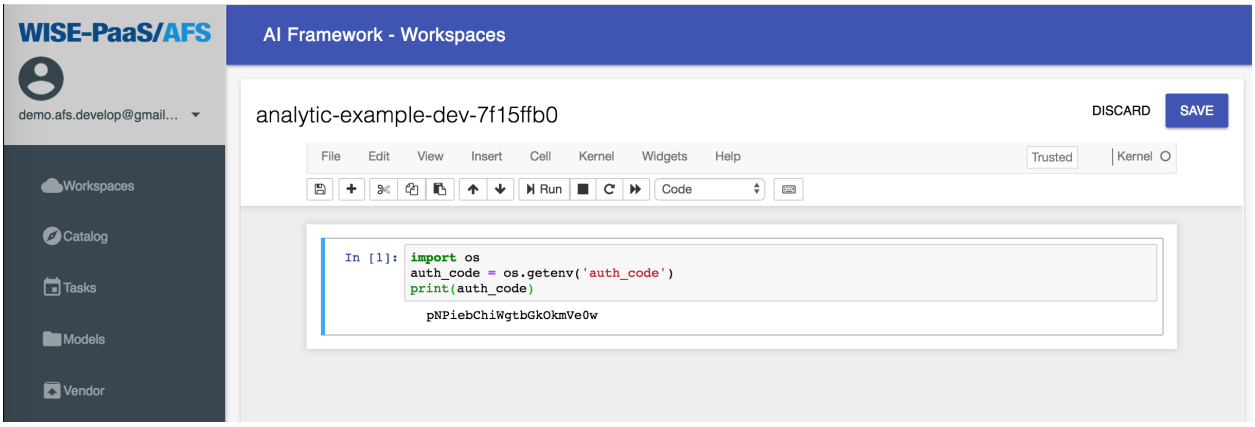
1.1.2 auth_code

The `$auth_code` is an environment variable from **Online Code IDE**, and the purpose of `$auth_code` is authenticating with **AFS** to use **AFS** functions in your analytics.

To check `$auth_code` of **Online Code IDE**, you can use the following snippet:

```
import os
auth_code = os.getenv('auth_code')
print(auth_code)
```

The output:

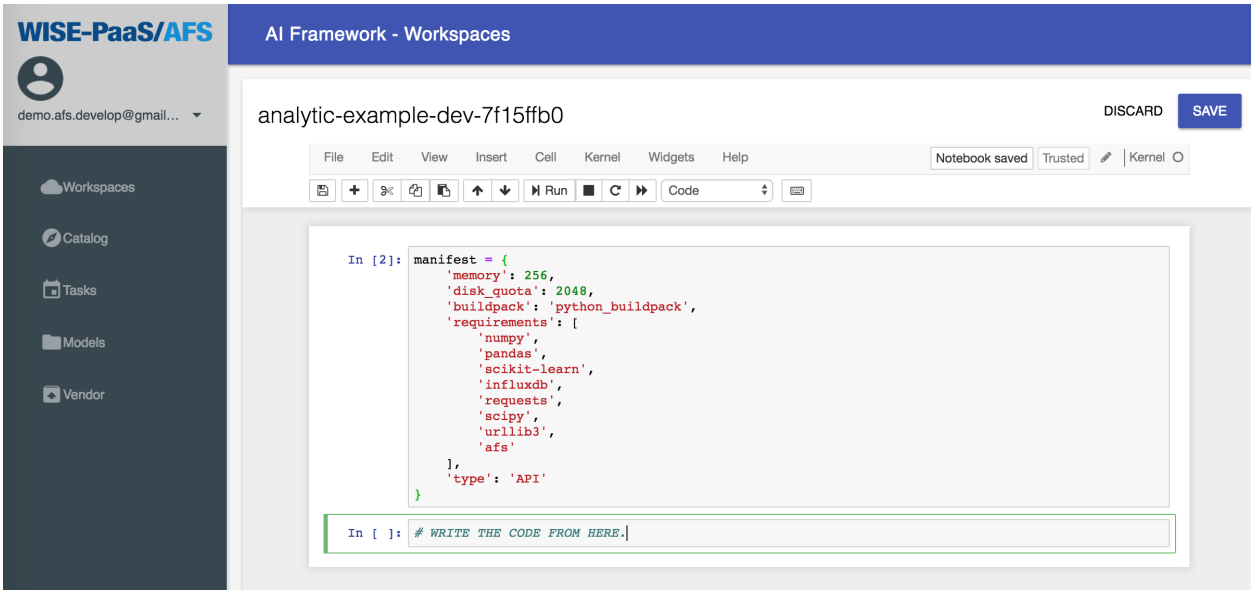


1.1.3 Manifest

In **Online Code IDE**, you can define some customize configurations like **memory**, **disk**, or **requirements** for your analytic by declaring a **manifest** at the first cell. When coding the analytics, and need to use the **AFS SDK** package, we can add the required package in the "requirements" of the manifest.

An example is as follows:

```
manifest = {
    'memory': 256,
    'disk_quota': 2048,
    'buildpack': 'python_buildpack',
    'requirements': [
        'numpy',
        'pandas',
        'scikit-learn',
        'influxdb',
        'requests',
        'scipy',
        'urllib3',
        'afs'
    ],
    'type': 'API'
}
```



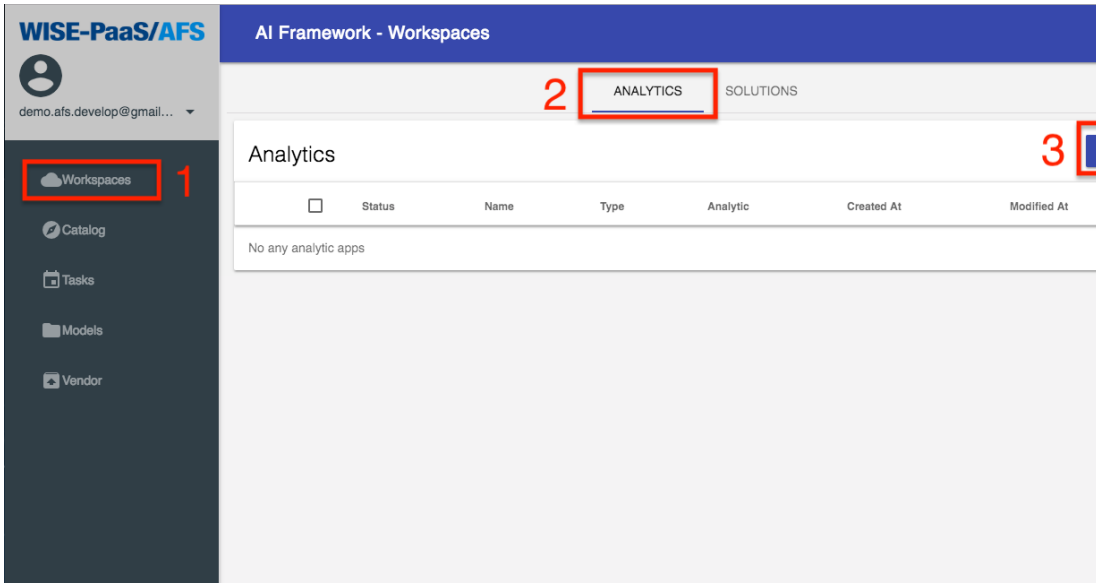
In this example, **memory** and **disk_quota** are also assigned to 2048MB. If set **memory** or **disk_quota** as **int** type, the default unit is **MB**. Or, use **str** type and you can specify the unit in **M**, **MB**, **G**, or **GB**.

> **Note:** The default value of **disk_quota** is **2048MB** to avoid insufficient disk space when installing modules. If you set **disk_quota** less than 2048MB, the value will be overridden to 2048MB.

The **requirements** are the most important part in analytic develop. As native python develop, when you need some external modules, you can use **requirements.txt** to record all dependencies of your analytic. (More information can be found at pip docs.) Provide a list of **requirements** can obtain the same effect when developing analytic by **AFS**.

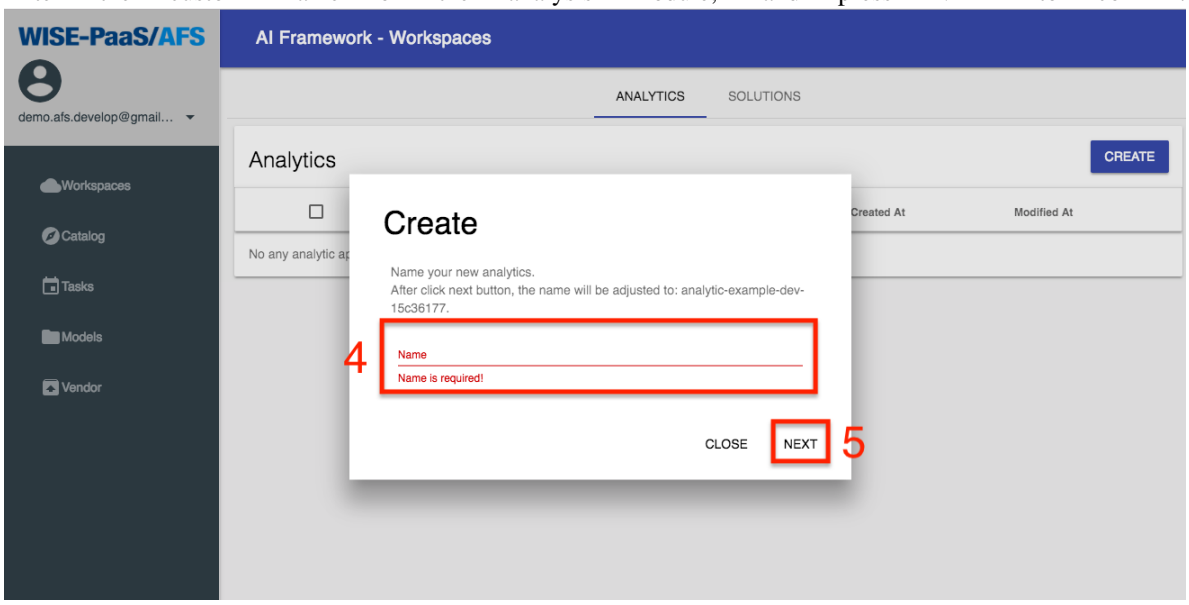
The **Type** is used to declare this analytic is an **APP** or an **API**. In default, all analytic will be assigned as **APP** type. But if you want your analytic serve as an **API** (and also write in any web framework), you need set **type** to **API** to host your analytic on **WISE-PaaS**.

1.1.4 Create analytic with Online Code IDE



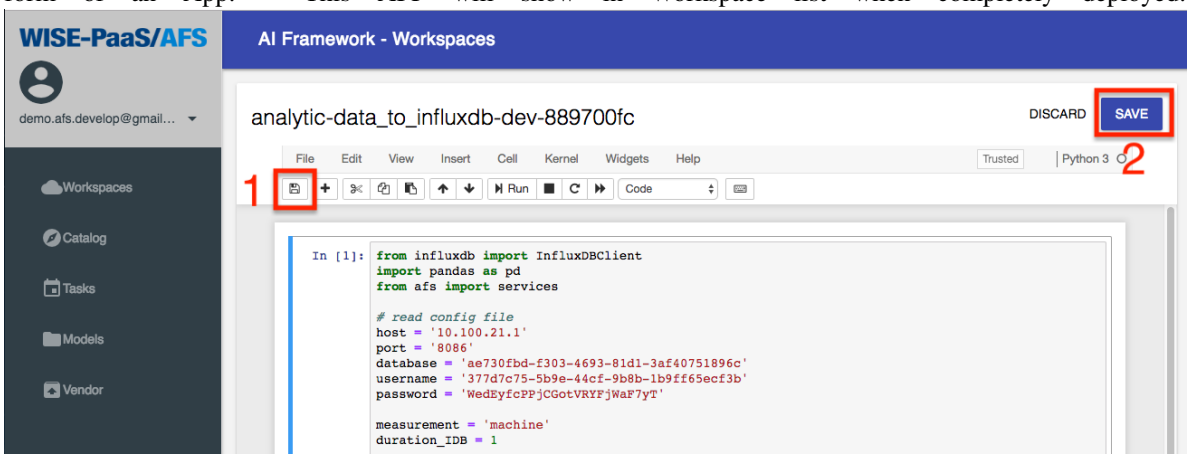
1. Click the `CREATE` button.

2. Enter the custom name of the analysis module, and press NEXT to confirm.



3. When the newly established development & editing page appears in the workspace, it means the module has been successfully created and you can write the analysis module using Python programming language.

4. After filling in the program code, you can click the icon to save it. Next, click `SAVE` button to push analysis training model application to the platform in the form of an App. This APP will show in Workspace list when completely deployed.



1.1.5 Install module with Vendor in private cloud

In python develop, we can use `pip install $MODULE` to install all required module. But in a private cloud, there is no any external internet resource can be used, including PyPI.

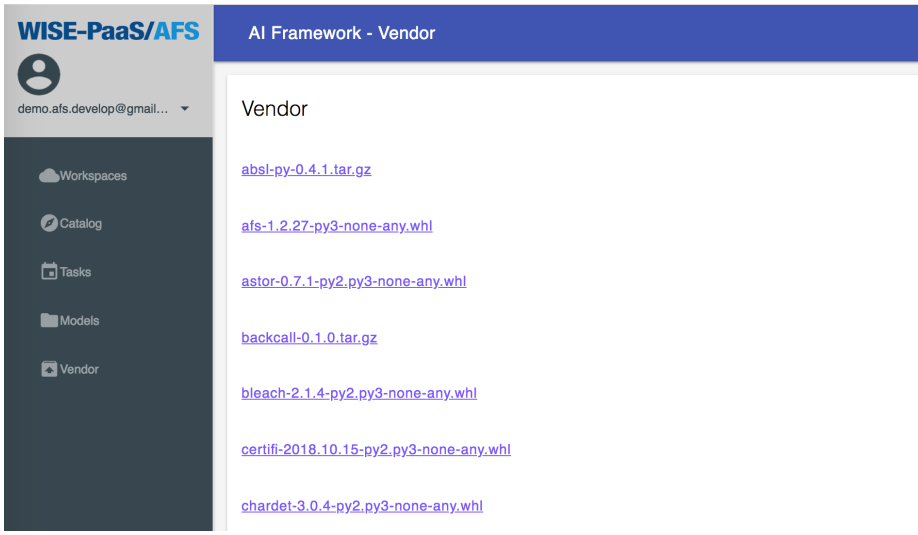
This restrict force all required modules should provide an offline distribution file in the private cloud when developing in **Online Code IDE** and save the source code to an analytic app.

This section will provide an example to use **Vendor** of AFS to install a module in **Online Code IDE**. Assume the module is already uploaded to **AFS**, if not, please reference documentation of Vendor to upload module.

- Note: When using **Vendor** of AFS to install a module in **Online Code IDE**, we must add the module in the

requirements of **manifest**, please refer the manifest section.

1. Right-click on the module and copy the url.



2. In **Online Code IDE**, use the following command and paste copied module url to install modules from the vendor:

```
! pip install $MODULE_URL?auth_code=$auth_code
```

1.1.6 Example of Online Code IDE

Here is an example to create Analytic API by Online Code IDE.

Step 1: Create a new Online Code IDE, please name it training_dt_model. About the detail, please refer to the **Create analytic with Online Code IDE** section.

Step 2: Declare the manifest. Declaring the **manifest** at the first cell. About the detail, please refer to the **Manifest** section.

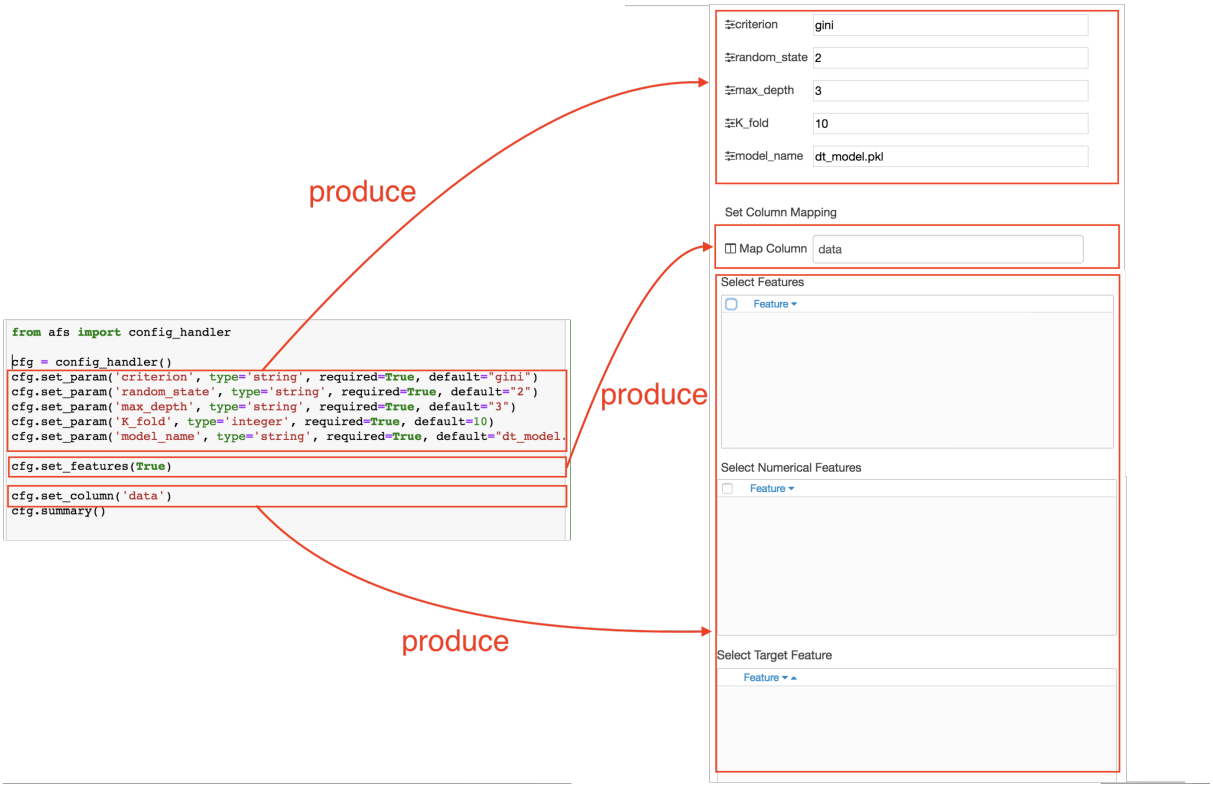
```
manifest = {
    'memory': 1024,
    'disk_quota': 2048,
    'buildpack': 'python_buildpack',
    'requirements': [
        'numpy',
        'pandas',
        'scikit-learn',
        'influxdb',
        'requests',
        'scipy',
        'urllib3',
        'afs'
    ],
    'type': 'API'
}
```

Step 3: Setting parameter of the analytic method. (We use the decision tree method for the example) In **Online Code IDE**, you can create a node on **Node-RED** by **SDK**, and you can provide the **Hyper-Parameter Tuning** for user. The following code must be at **second cell**.

```
from afs import config_handler
cfg = config_handler()
cfg.set_param('criterion', type='string', required=True, default="gini")
cfg.set_param('random_state', type='string', required=True, default="2")
cfg.set_param('max_depth', type='string', required=True, default="3")
cfg.set_param('K_fold', type='integer', required=True, default=10)

cfg.set_param('model_name', type='string', required=True, default="dt_model.pkl")
cfg.set_features(True)
cfg.set_column('data')
cfg.summary()
```

- Note: When editing is complete in this cell, you must run it.

Describe the features that the SDK can produce, here is an example of **Decision Tree**.



Step 4: Training model Here is an example of **Decision Tree**: import package:

```
from sklearn import tree
from sklearn.cross_validation import train_test_split
from sklearn import metrics
from sklearn.externals import joblib
from afs import models
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelBinarizer
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

import pandas as pd
```

(continues on next page)

(continued from previous page)

```
import numpy as np
import json
import requests
```

Defined function:

```
#Find the best parameter to training model
def grid(data, target, parameters_dt, cv):
    clf = tree.DecisionTreeClassifier()
    grid = GridSearchCV(estimator = clf, param_grid = parameters_dt, cv = cv,
                        scoring = 'accuracy')
    grid.fit(data,target)
    best_accuracy = grid.best_score_
    best_params = grid.best_params_
    return best_accuracy,best_params
```

```
#Take the best parameter to training model
def training_model(data, target,best_params, best_accuracy,model_name):
    clf = tree.DecisionTreeClassifier(**best_params)
    clf = clf.fit(data, target)
    #save model
    joblib.dump(clf, model_name)
    client = models()
    client.upload_model(model_name, accuracy=best_accuracy, loss=0.0,␣
↪tags=dict(machine='dt'))

    return model_name
```

Main program:

```
# POST /

# Set flow architecture, REQUEST is the request including body and headers from client
cfg.set_kernel_gateway(REQUEST)
# Get the parameter from Node-RED setting

criterion = str(cfg.get_param('criterion'))
random_state = str(cfg.get_param('random_state'))
max_depth = str(cfg.get_param('max_depth'))
cv = cfg.get_param('K_fold')


model_name = str(cfg.get_param('model_name'))
select_feature = cfg.get_features_selected()
data_column_name = cfg.get_features_numerical()
target2 = cfg.get_features_target()

labels_column_name = [x for x in select_feature if x not in data_column_name]
labels_column_name = [x for x in labels_column_name if x not in target2]

if(labels_column_name==[]):
    labels_column_name=["No"]

a1=["time"]
labels_column_name = [x for x in labels_column_name if x not in a1]
```

(continues on next page)

(continued from previous page)

```
if "All" in labels_column_name:
    labels_column_name.remove("All")

if(data_column_name==[]):
    data_column_name=["No"]

criterion = criterion.split(",")
random_state = random_state.split(",")
max_depth = max_depth.split(",")

random_state =list(map(int, random_state))
max_depth = list(map(int, max_depth))

parameters_dt = {"criterion": criterion, "random_state": random_state, "max_depth":␣
↪max_depth}



# Get the data from request, and transform to DataFrame Type
df = cfg.get_data()
df = pd.DataFrame(df)

target = np.array(df.loc[:,[target2]])

if (data_column_name[0]=="All"):
    all_df_column = [df.columns[i] for i in range(len(df.columns))]
    if (labels_column_name[0]!="No"):
        for i in range(len(labels_column_name)):
            all_df_column.remove(labels_column_name[i])
        all_df_column.remove(target2)
    if (labels_column_name[0]=="No"):
        all_df_column.remove(target2)
    data = np.array(df.loc[:,all_df_column])

elif (data_column_name[0]=="No"):
    data = np.array([]).reshape(df.shape[0],0)
    if (labels_column_name[0]!="No"):
        for i in labels_column_name:
            if ((False in map((lambda x: type(x) == str), df[i]))==False):
                label2 = LabelBinarizer().fit_transform(df[i])
                data = np.hstack((data,label2))
            if ((False in map((lambda x: type(x) == int), df[i]))==False):
                target9 = OneHotEncoder( sparse=False ).fit_transform(df[i].values.
↪reshape(-1,1))
                data = np.hstack((data,target9))

else:
    data = np.array(df.loc[:,data_column_name])
    if (labels_column_name[0]!="No"):
        for i in labels_column_name:
            if ((False in map((lambda x: type(x) == str), df[i]))==False):
                label2 = LabelBinarizer().fit_transform(df[i])
                data = np.hstack((data,label2))
            if ((False in map((lambda x: type(x) == int), df[i]))==False):
                target9 = OneHotEncoder( sparse=False ).fit_transform(df[i].values.
↪reshape(-1,1))
                data = np.hstack((data,target9))
```

(continues on next page)

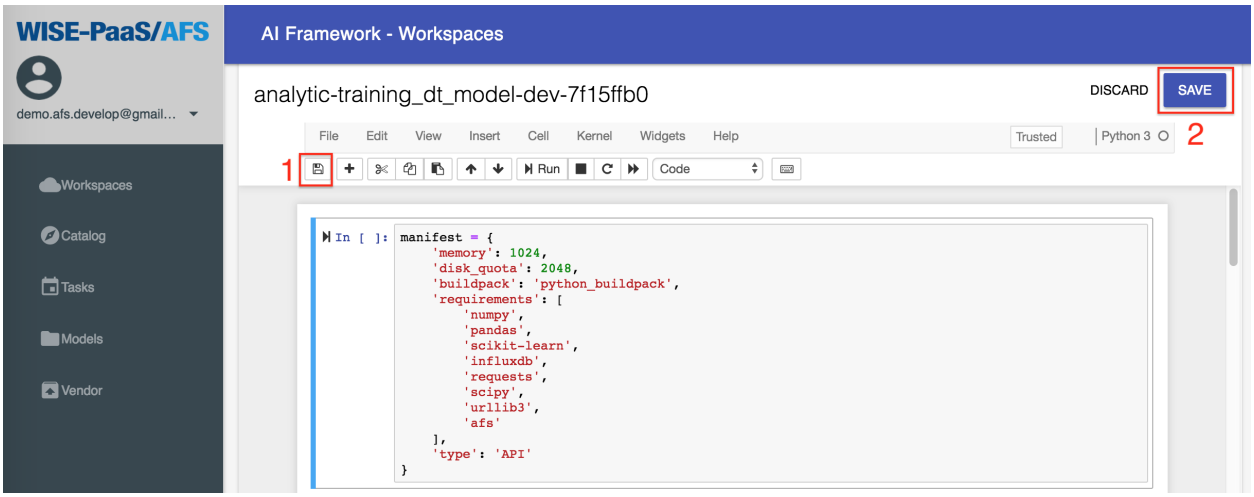
(continued from previous page)

```
best_accuracy,best_params = grid(data, target, parameters_dt, cv)
result = training_model(data, target,best_params, best_accuracy,model_name)
result = str(result)

df2 = pd.DataFrame([result], columns=['model_name'])
# df_dict = df2.to_dict()


# # Send the result to next node, and result is  DataFrame Type

ret = cfg.next_node(df2, debug=False)

# # The printing is the API response.
print(json.dumps(ret))
```

Step 5: Save and upload the Analytic API After we edit the Analytic App, we must save and upload it as follow steps:

(i) Click the icon  is in upper left corner.

(ii) Click `SAVE`, and we are uploading the Analytic App now.

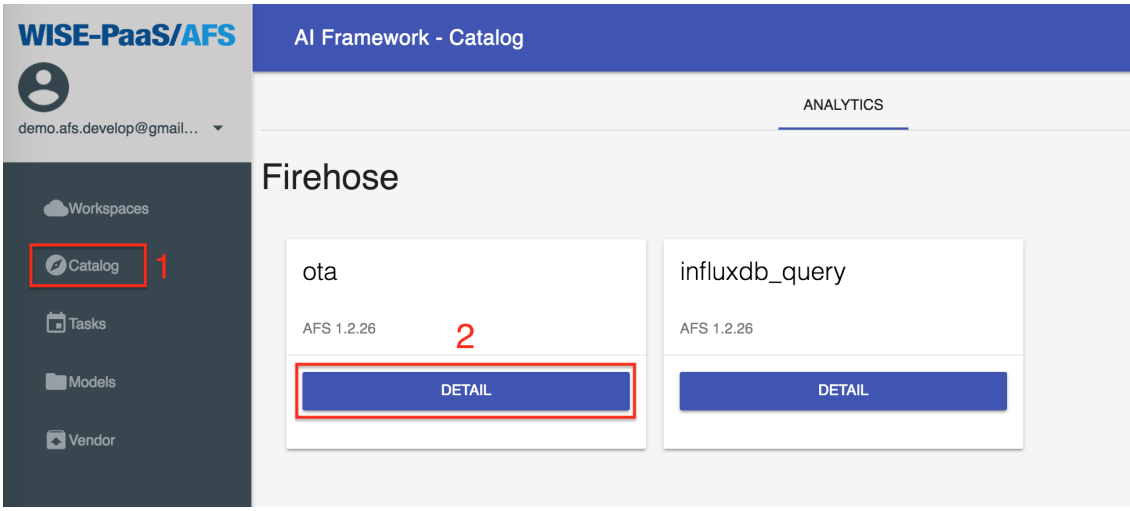


Wait a second, we can see that it's successful to upload.

1.2 Solution

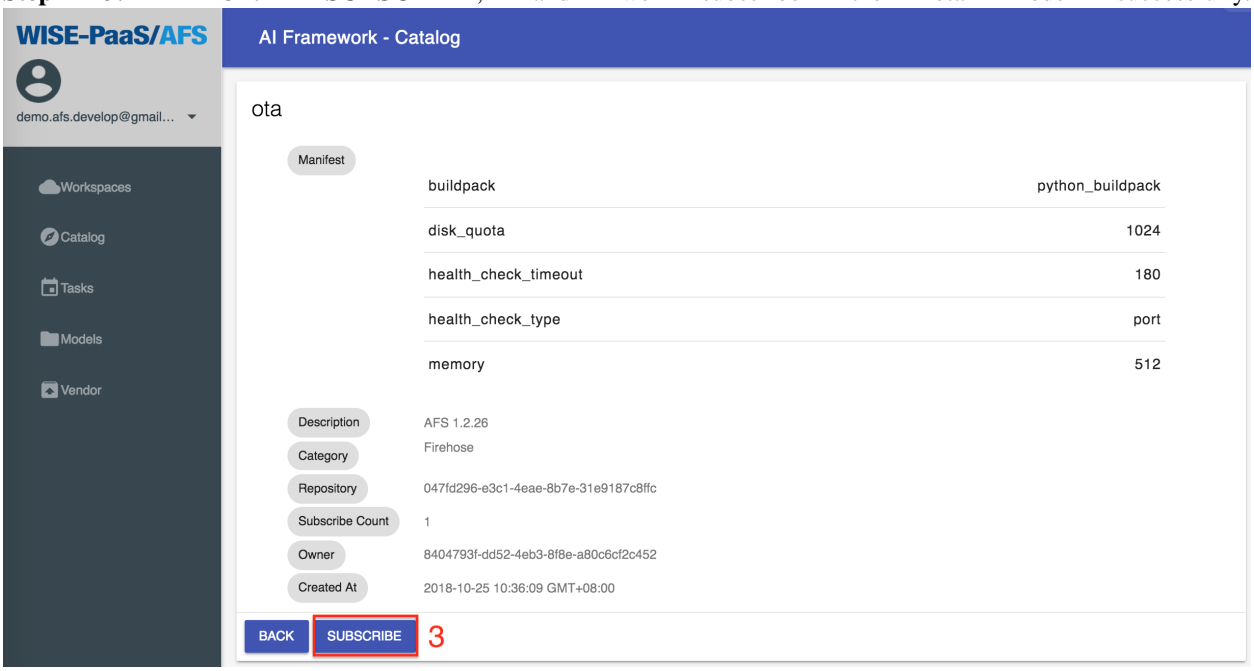
Pre-condition: Before creating a solution, there are preparations we must get ready. In the beginning, subscribing **ota** node and influxdb_query node from Catalog is required. Now, we subscribe the ota node firstly.

Step 1: Click **Catalog**.



Step 2: Click ota's **DETAIL**.

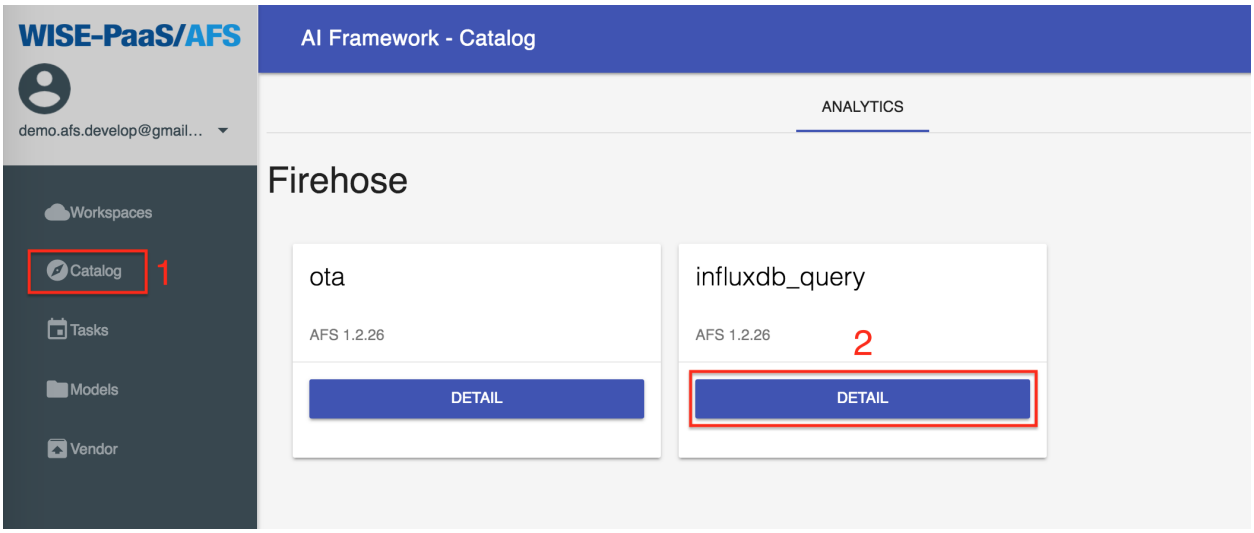
Step 3: Click **SUBSCRIBE**, and we subscribe the ota node successfully.



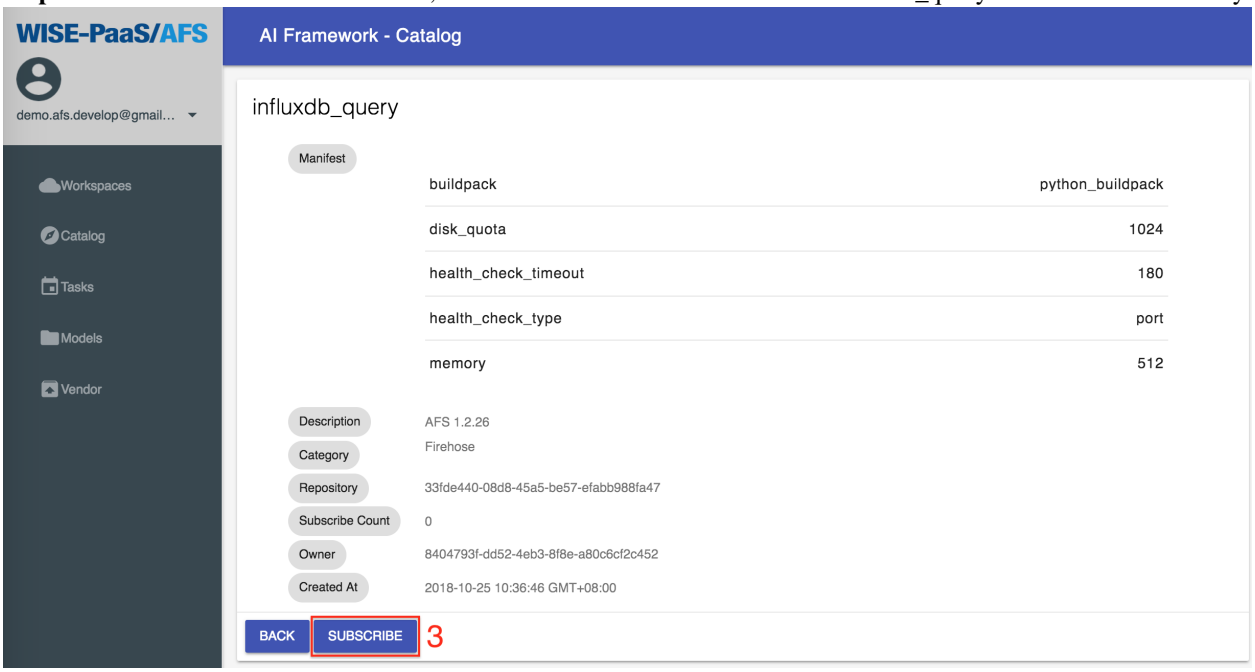
Next, we subscribe the firehose node.

Step 4: Click **Catalog**.

Step 5: Click influxdb_query's **DETAIL**.



Step 6: Click **SUBSCRIBE**, and we subscribe the influxdb_query node successfully.



Step 7: Click **Workspaces**, go back to workspace.

After subsribing the nodes, the system will rederect to the Analytics page. Wait a second, the Analytic APIs are created successfully.

1.2.1 Creating a new solution instance

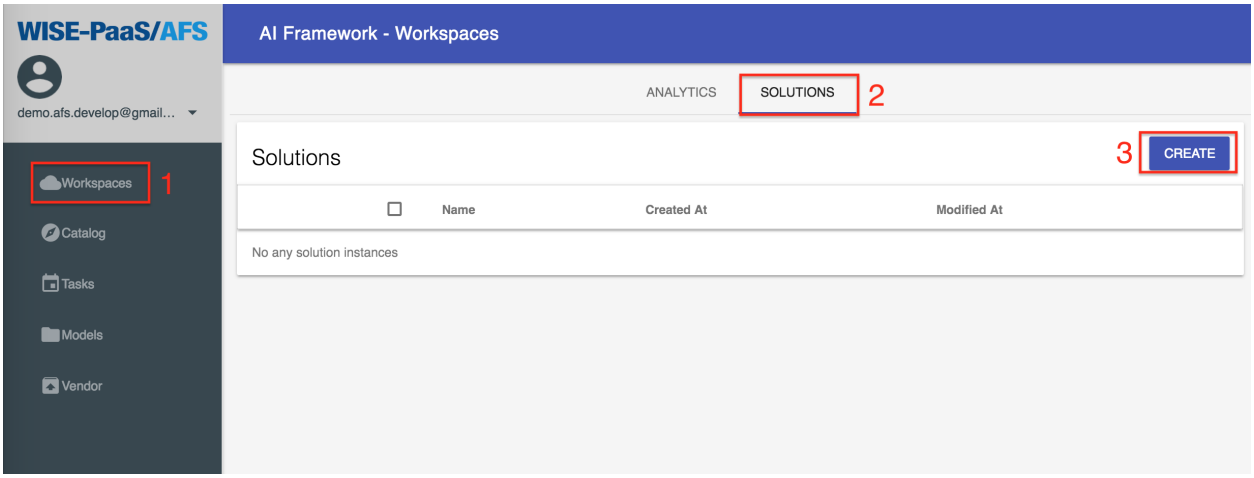
Now, we start to create a new solution.

There are the steps as follows:

Step 1: Click **Workspaces**.

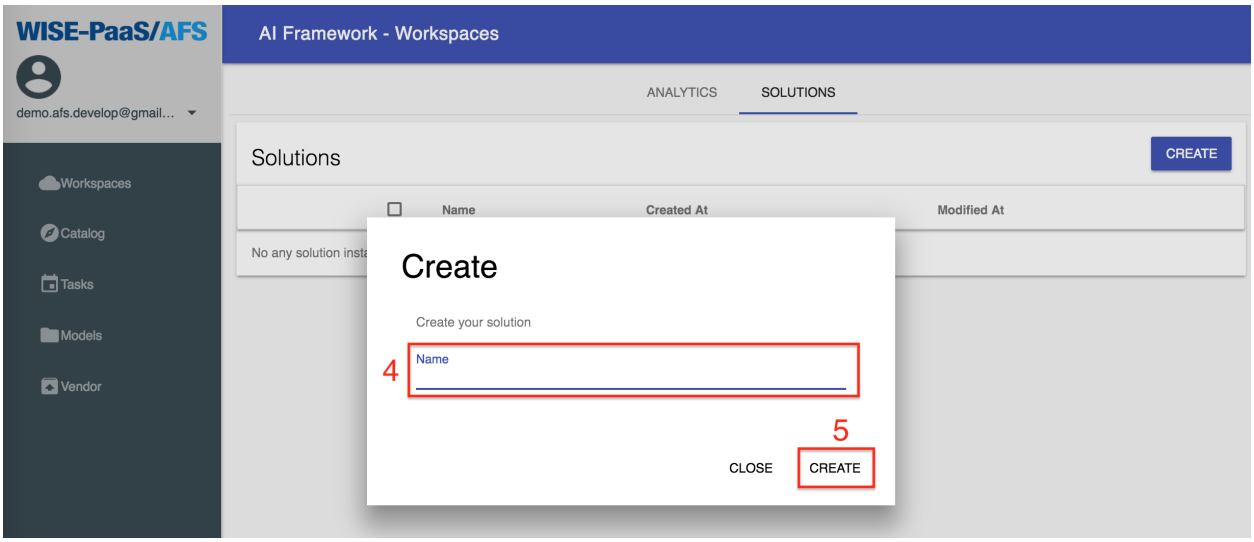
Step 2: Click **SOLUTIONS**.

Step 3: Click **CREATE**.

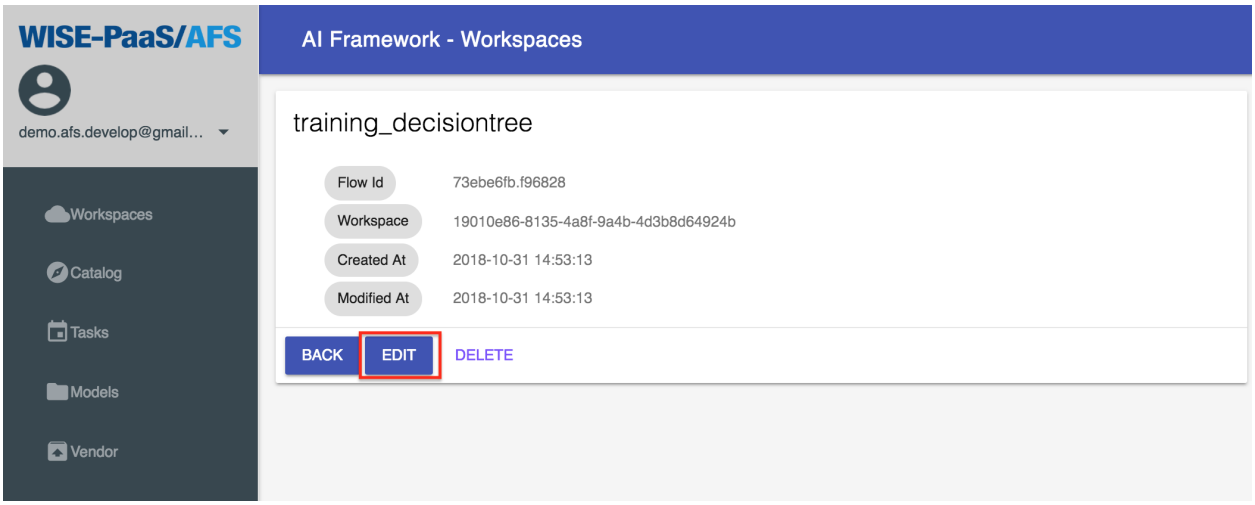


Step 4: Enter the solution name.

Step 5: Click **CREATE** to create the solution.



Step 6: Click **EDIT**, and the online flow IDE is shown, and we can start to create the flow.



1.2.2 Start create the solution by Online Flow IDE

In the **Pre-condition** step, we create ota node and influxdb_query node. As the example in **Example of Code IDE**, we create a Decision Tree node. The sso_setting already exists. Now, we have **sso_setting** node, **influxdb_query** node, **training_dt_model** node, and **ota** node.

- How to create **training_dt_model** node, please refer **Example of Online Code IDE** above.

You need pull four nodes such that **sso_setting**, **influxdb_query**, **training_dt_model**, and **ota**.

Setup the nodes

1. The **sso_setting** node

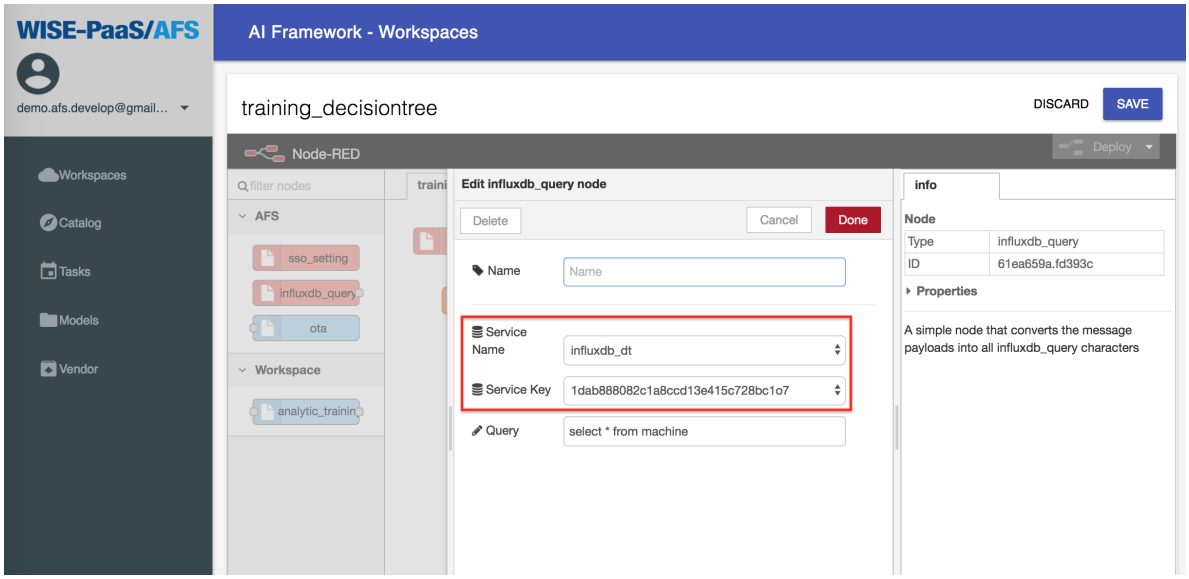
   Step 1: Enter **SSO User** and **SSO Password**.

   Step 2: Click **DONE** to save and exit the setting.

2. The **firehose_influxdb_query** node

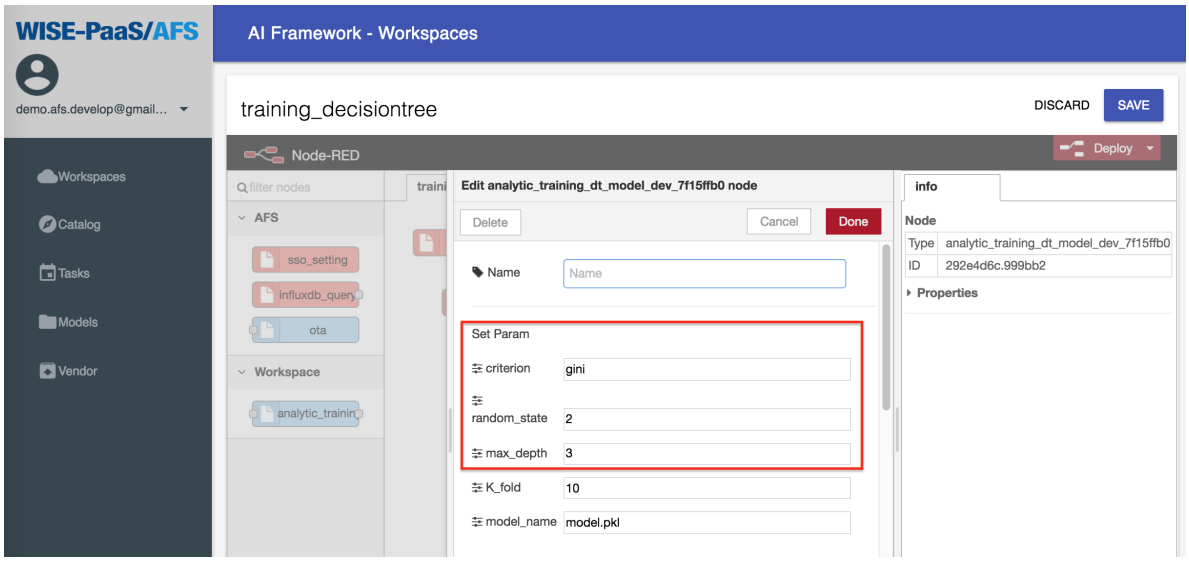
   Step 1: Choose **Service Name**, **Service Key**, and enter **Query** condition.

   Step 2: Click **DONE** to save your setting.



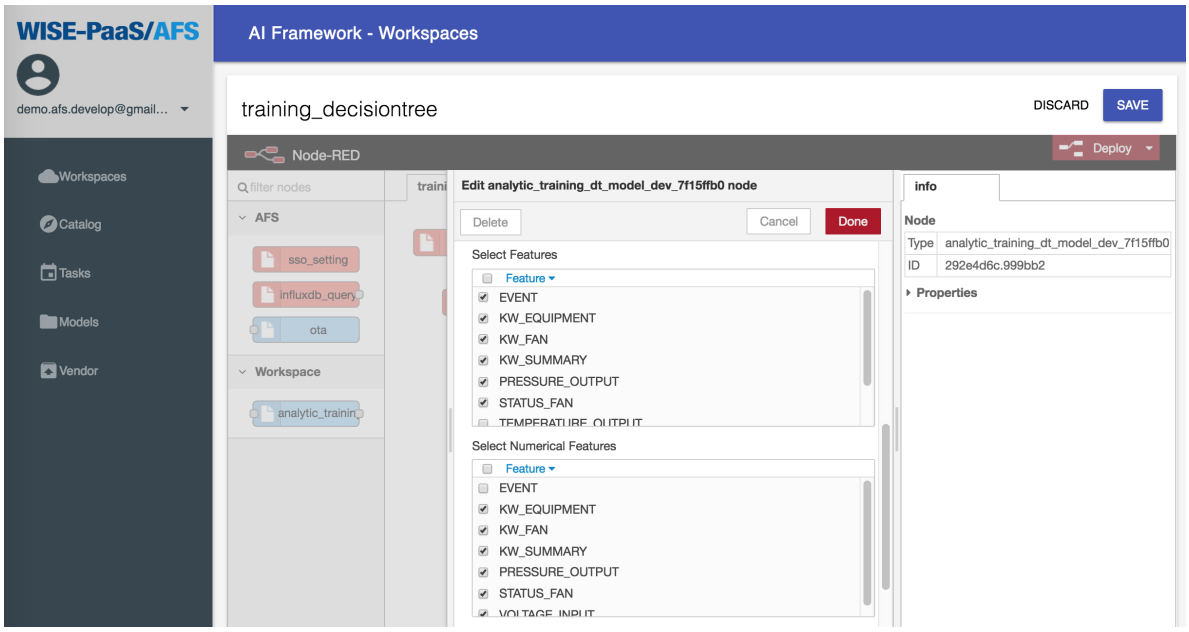
3. The **training_dt_model** node

   Step 1: Enter **parameters** to training model.



Step 2: Select **features** to training model.

Step 3: Select **numerical features**.



Step 4: Select **target features** to training model.

Step 5: Please click **DONE** to save your setting when you complete the setup.

4. The **ota** node

   Step 1: Choose **Device Name** and **Storage Name**.

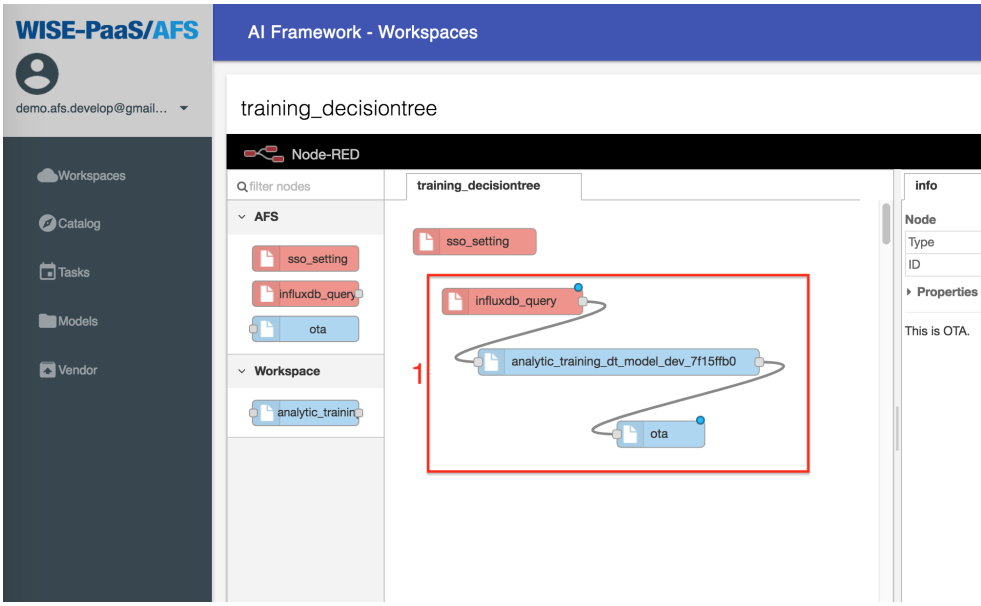
   Step 2: Please click **DONE** to save the setting when you complete the setup.

5. **Nodes connecting**

   Step 1: Connect nodes, influxdb_query connection training_dt_model and training_dt_model connection ota that like the image below.

   Step 2: Click **Deploy** to save **Node-RED**.

Step 3: Click **SAVE** to save solution.

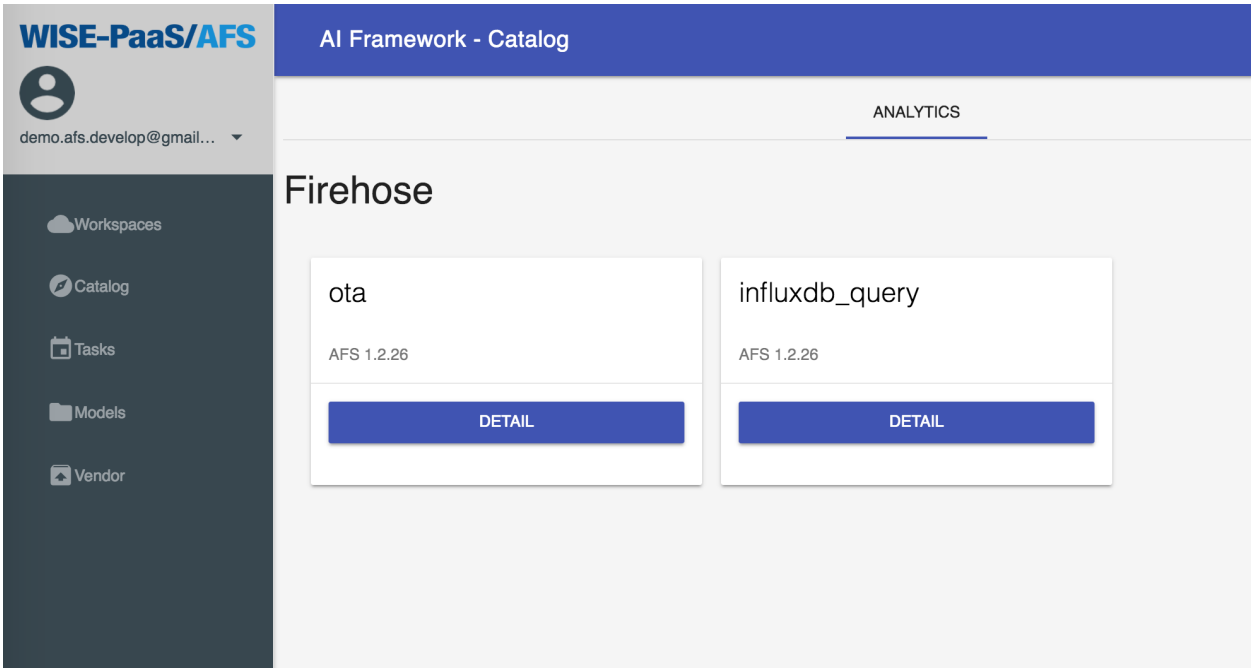


We create the solution successfully when it shows **Update complete** in the bottom right.

CHAPTER 2

Catalog

In **AFS**, we provide the analytic methods and tools in **Catalog**. The users can subscribe the methods and use them in **Workspaces**. When creating a **Solution** in **Workspaces**, there are two nodes, ota and influxdb_query, we must subscribe them then develop the new solution. More about creating new **Solution**, please refer Solution.



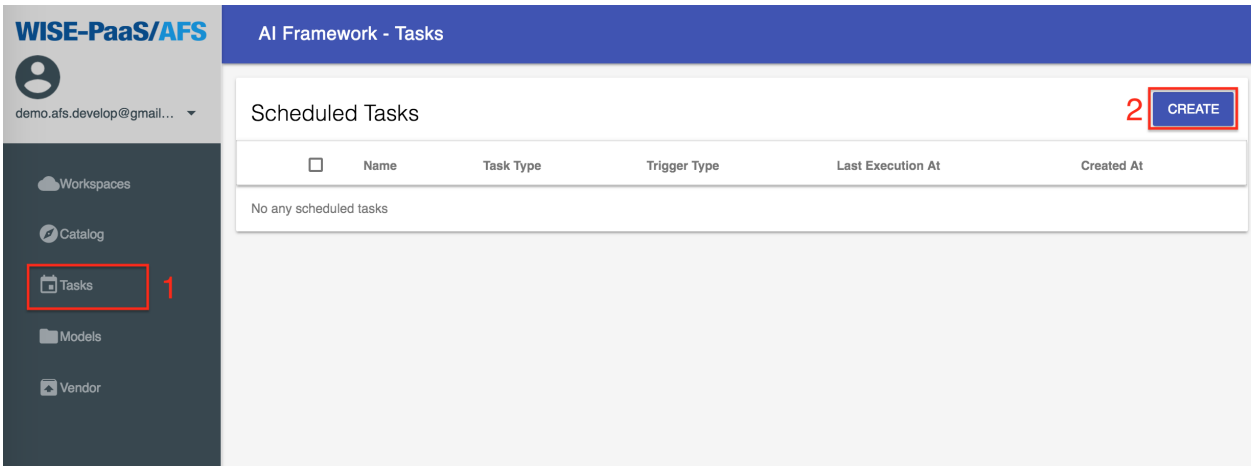
CHAPTER 3

Tasks

3.1 Create new task

Step 1: Click **Tasks**.

Step 2: Click **CREATE**.

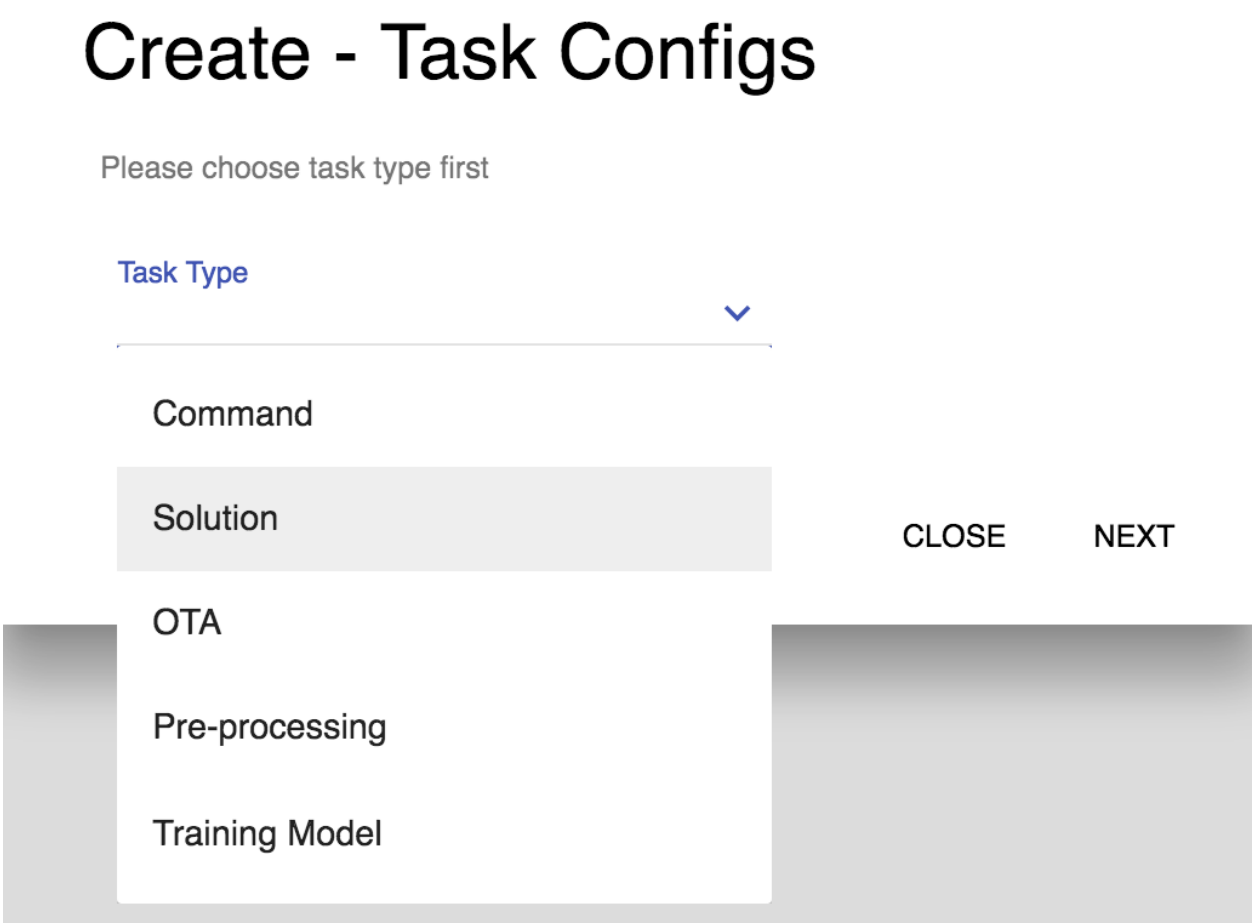


Step 3: Enter task name training_model.

Step 4: Click **NEXT**.

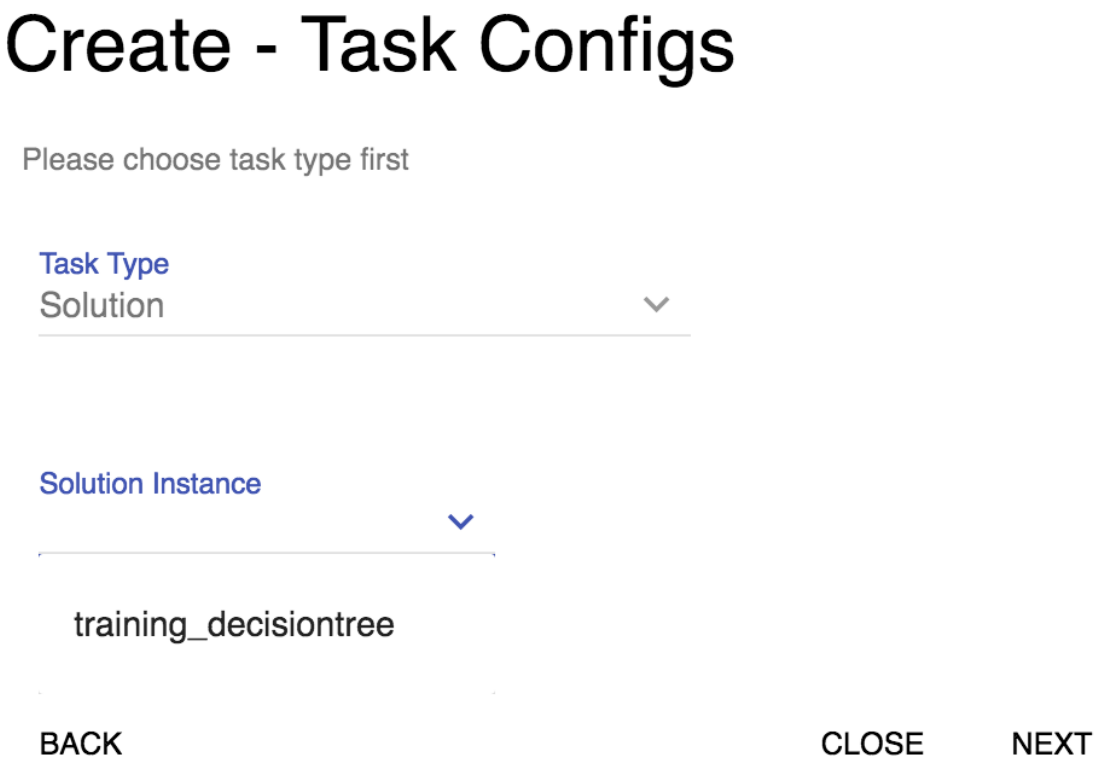
3.1.1 Task types

Step 5: Choose **Task Type**, then choose Solution.



Step 6: Choose **Solution Instance**. (You can choose which you create solution, please refer Solution to create your solution.)

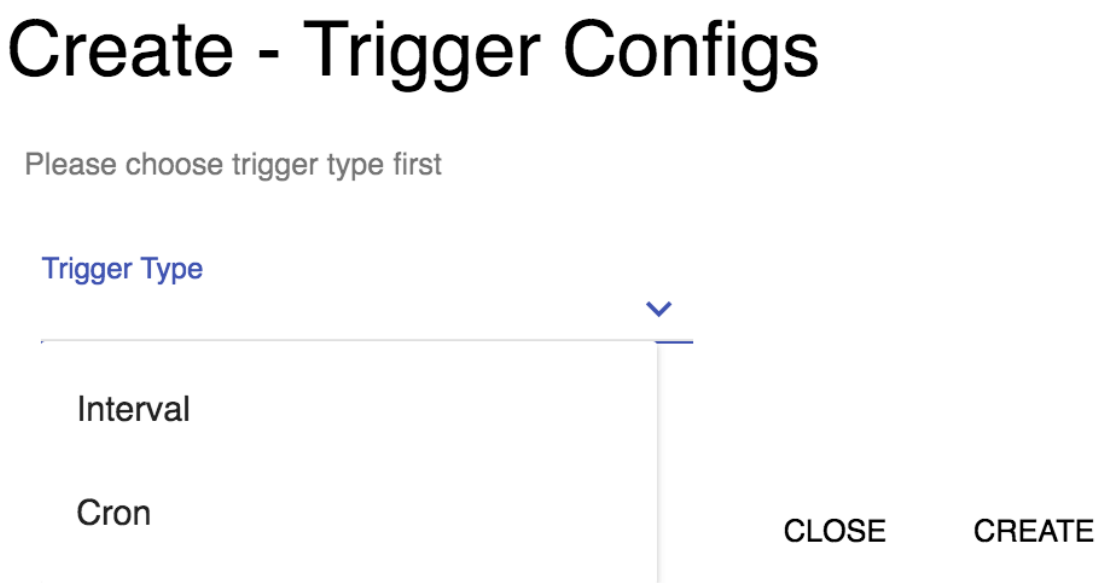
Step 7: Click **NEXT**.



3.1.2 Trigger types

Step 8: Choose **trigger type**.

In this example, we choose **Interval**.



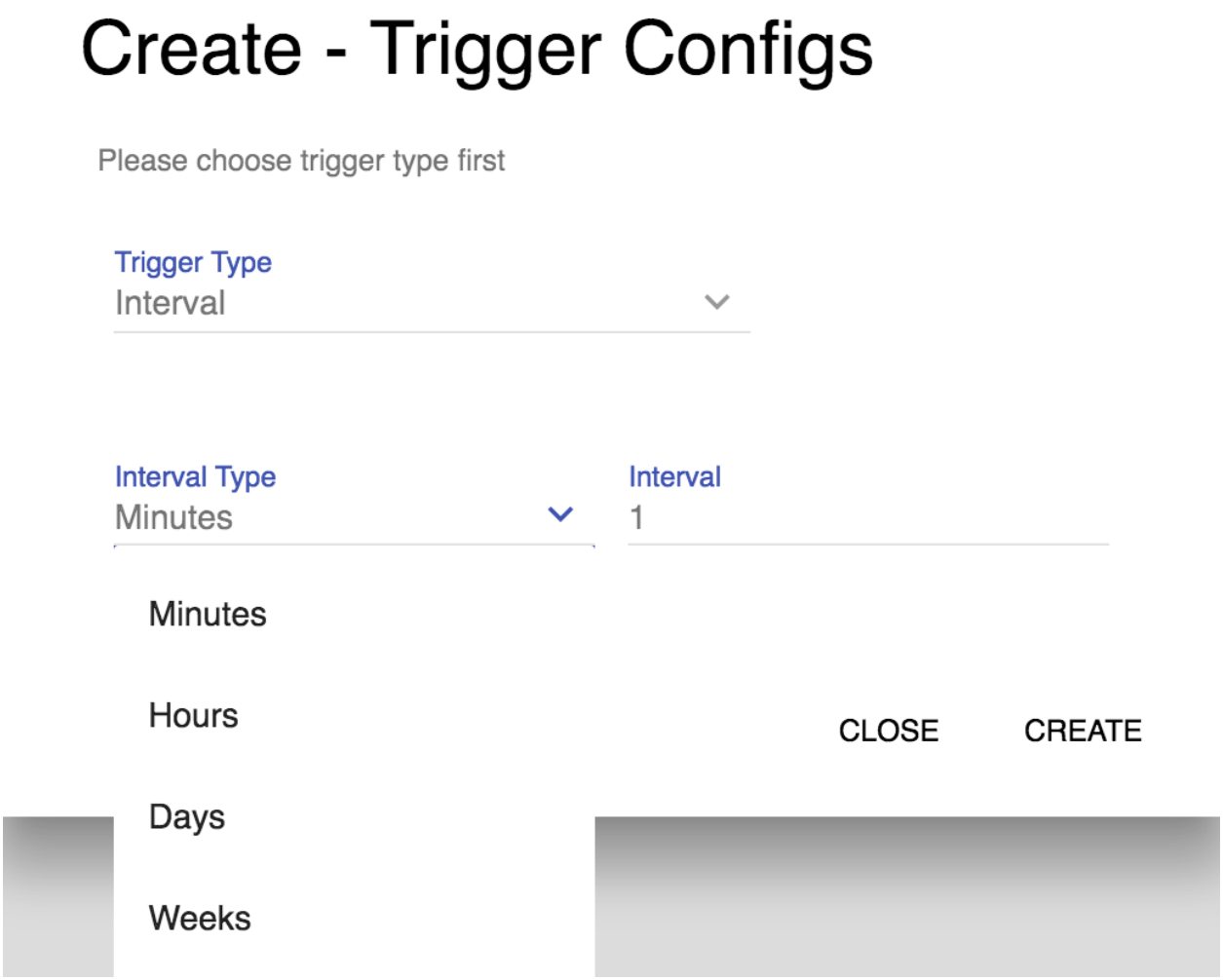
Step 9: Choose **Interval type**.

In this example, we choose **Minutes**.

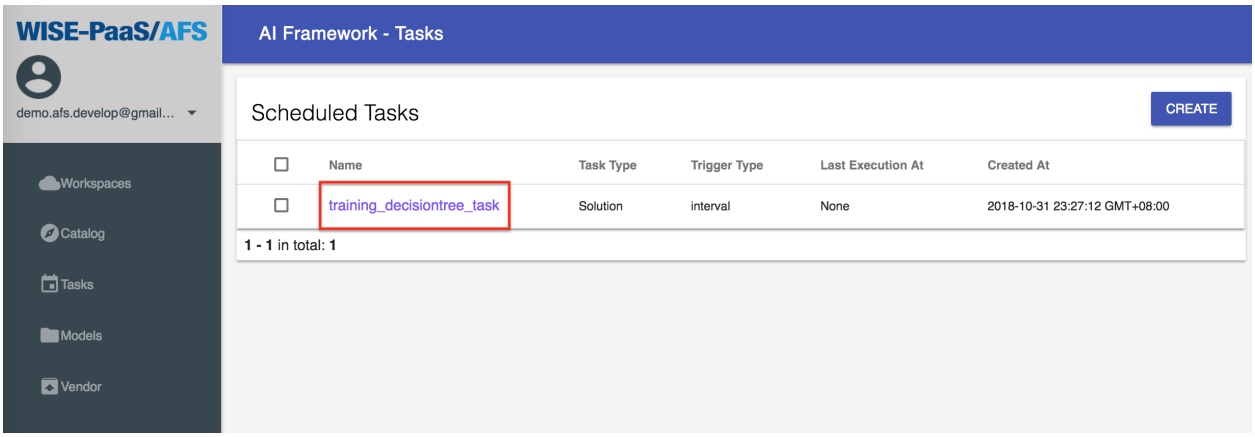
Step 10: Enter **Interval**.

In this example, we enter **1**.

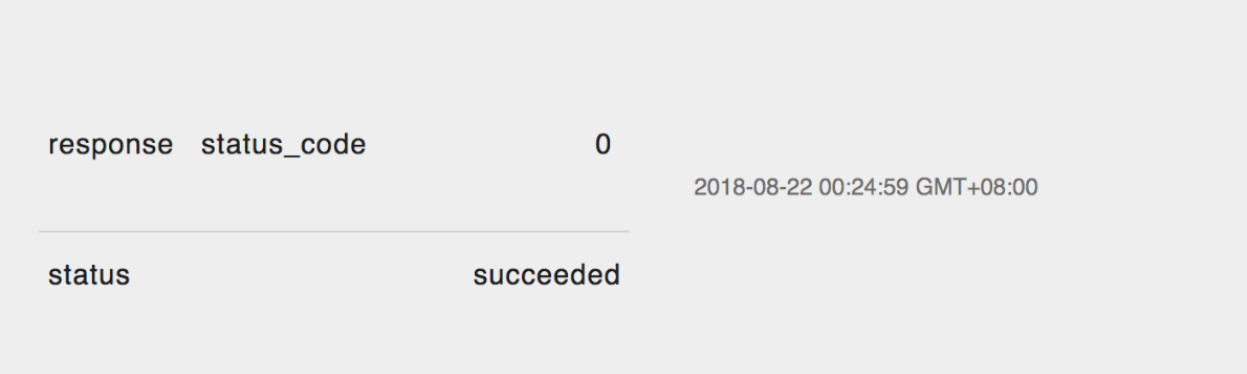
Step 11: Click **CREATE**.



Step 12: Click training model.



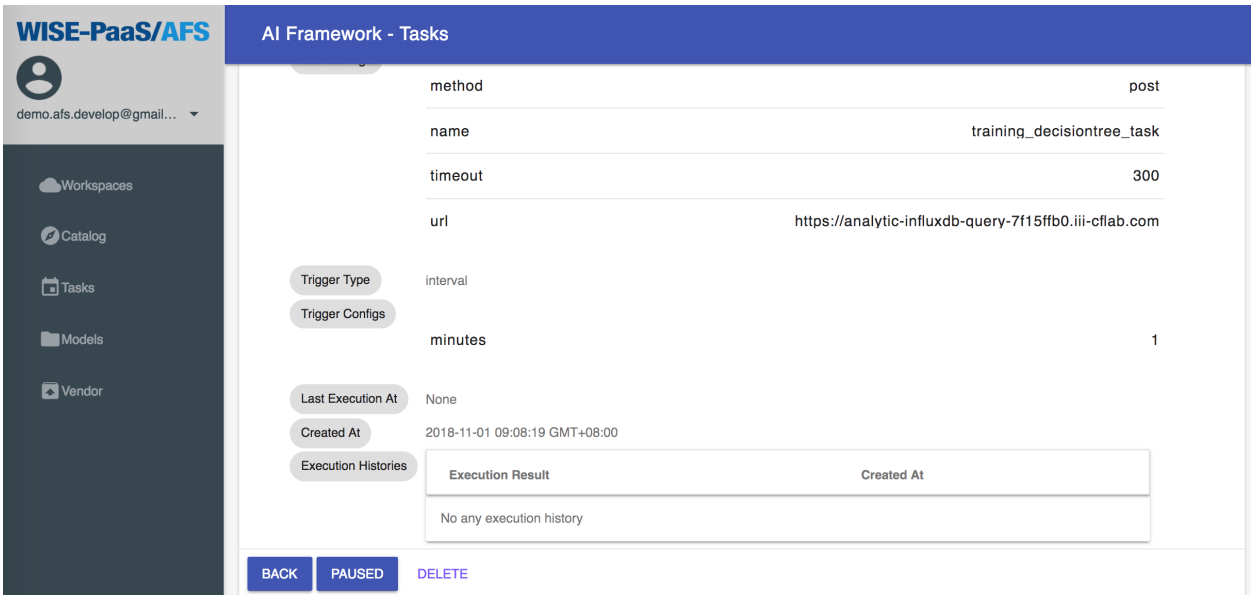
When the task has been executed, you can see like this.



- If timeout occurs, please adjust the interval size because the training time is greater than interval; **API** can only accept one request at a time, receiving multiple requests at a time will timeout.

3.2 More Task's Operations

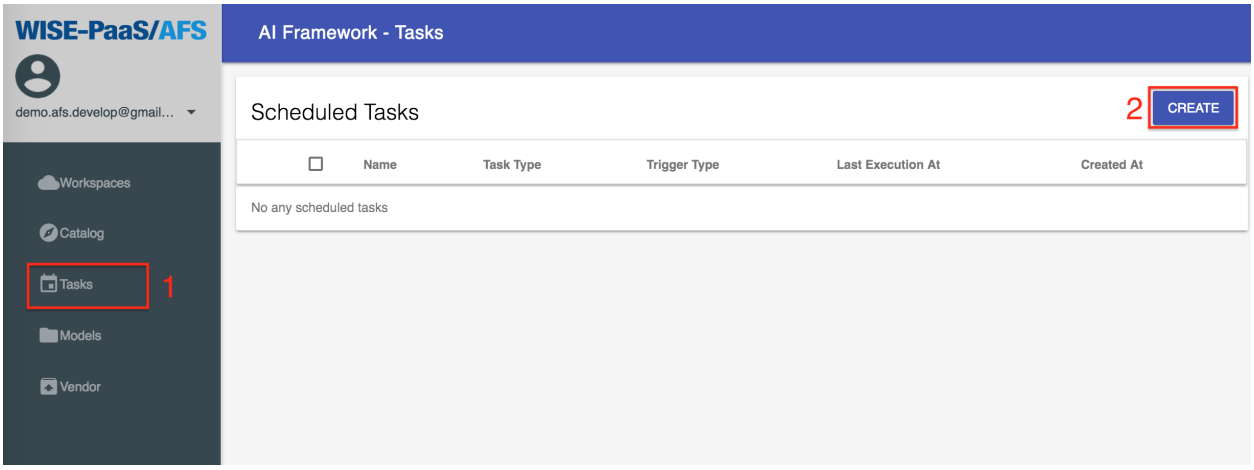
The users can operate the tasks by the requirement. There are three operations which are provided, include `PAUSE`, `RESUME`, and `DELETE`.



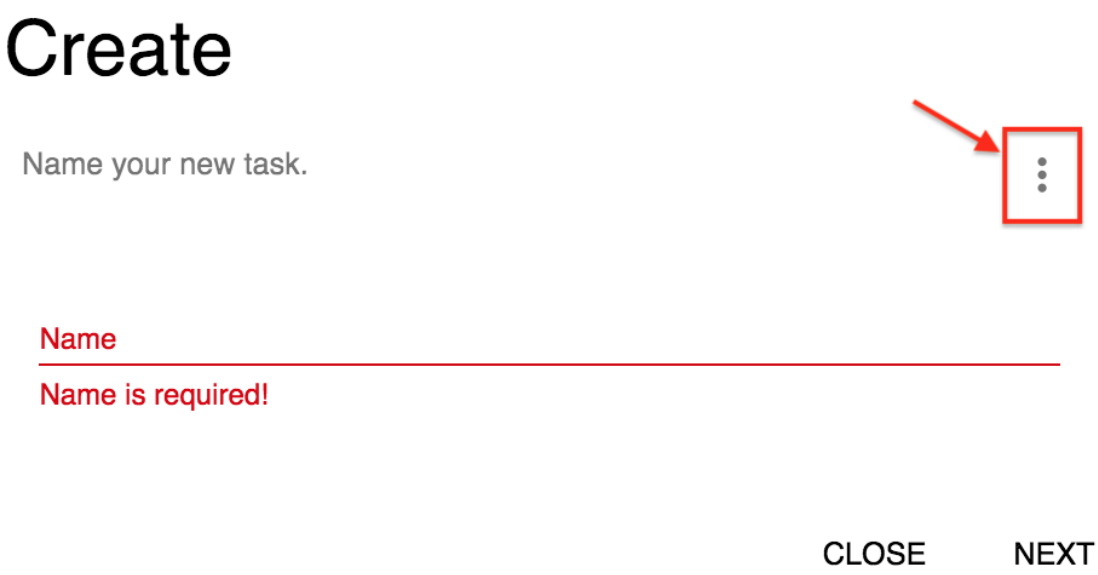
3.3 Create multiple tasks

Step 1: Click **Tasks**.

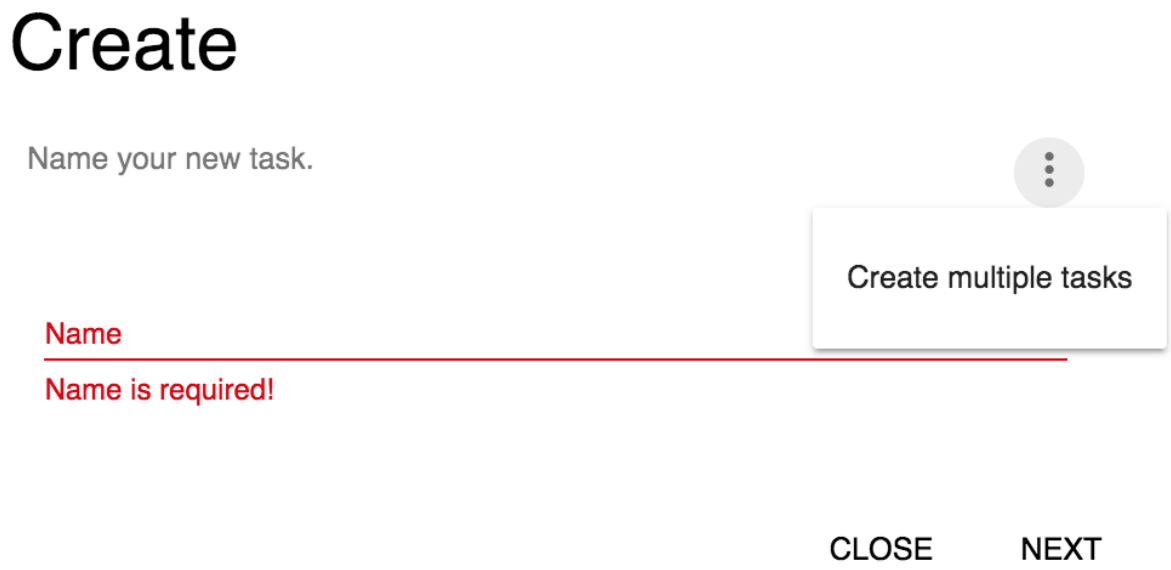
Step 2: Click **CREATE**.



Step 3: Click icon as follows.



Step 4: Click **Create multiple task**.



Step 5: Click **csv example** to download csv example.

Step 6: Choose csv file. (In the example, you must create APP first. Please refer Analytics).

- In the example, the parameter of **Cron** that refer the link.

Step 7: Click **CREATE**, and the tasks are created successfully.

3.4 Limitation

There are currently 5 threads executing tasks on the Kernel Gate Way. When it takes 3 minutes to train the model once, but the setup of the task is that execute every 1 minute, there will be an error condition in the task. Users must evaluate the schedule execution time for the task.

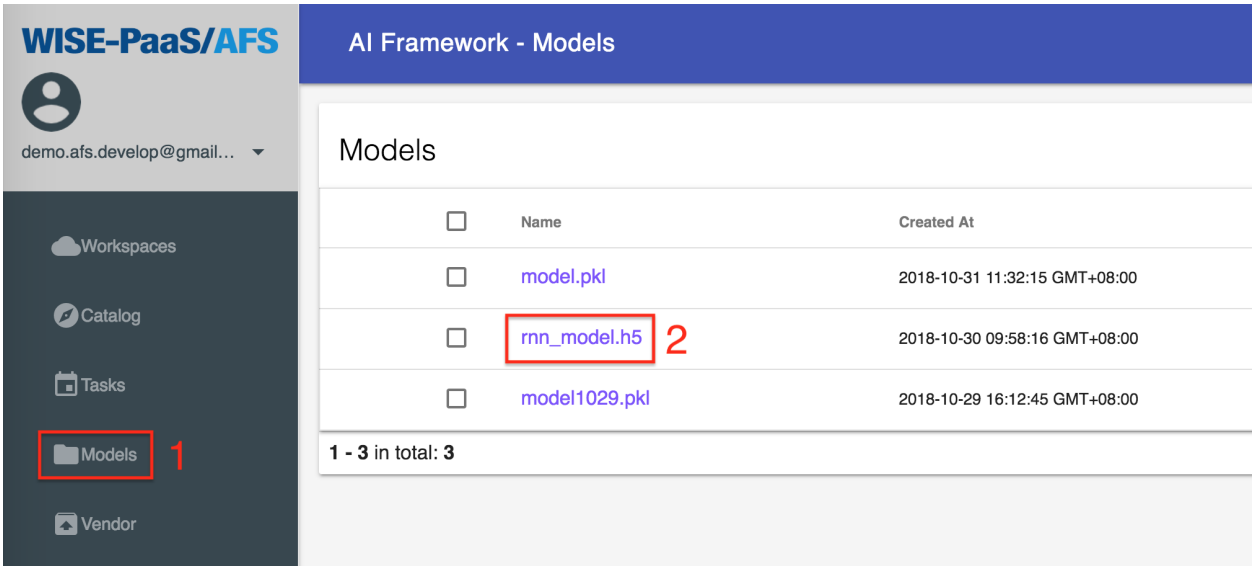
If the task does not work on the schedule, users can check the log in the online code IDE. Please refer the troubleshooting to see more details.

CHAPTER 4

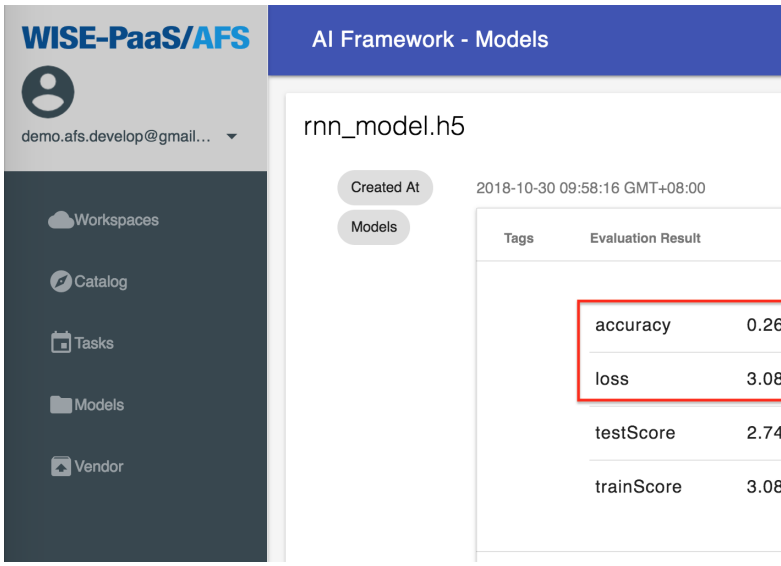
Models

After implementing the training APP, you can go to AFS Models. Select Repository of the model and read the performance value of the training result. About training model, please refer Example of Online Code IDE.The steps are as below:

1. Select the training APP after clicking Models.
2. The latest training time and performance value can be inquired.



Click the model to see the performance value of training result.

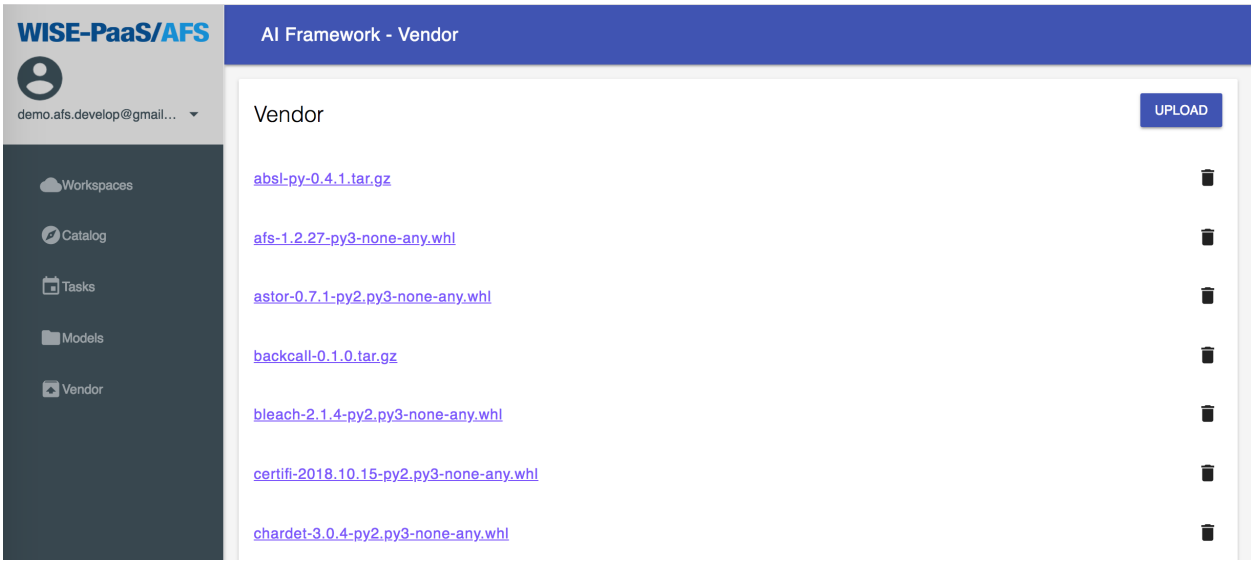


CHAPTER 5

Vendor

The **Vendor** provides modules support for private cloud version of AFS. This chapter will illustrate how to:

1. Download required module from PyPI.
2. Upload module to Vendor.
3. Delete module in Vendor.



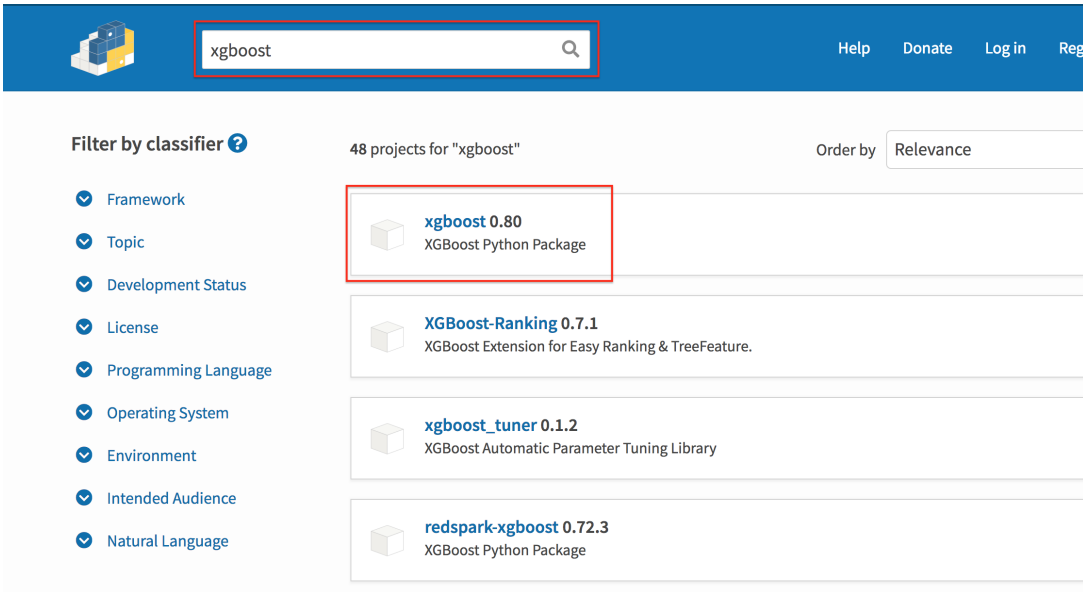
5.1 Download required module from PyPI

All analytic app in AFS will use **Python 3.6.x** on **Linux** as default runtime. If you want to add a new module to Vendor, please make sure version compatible of module.

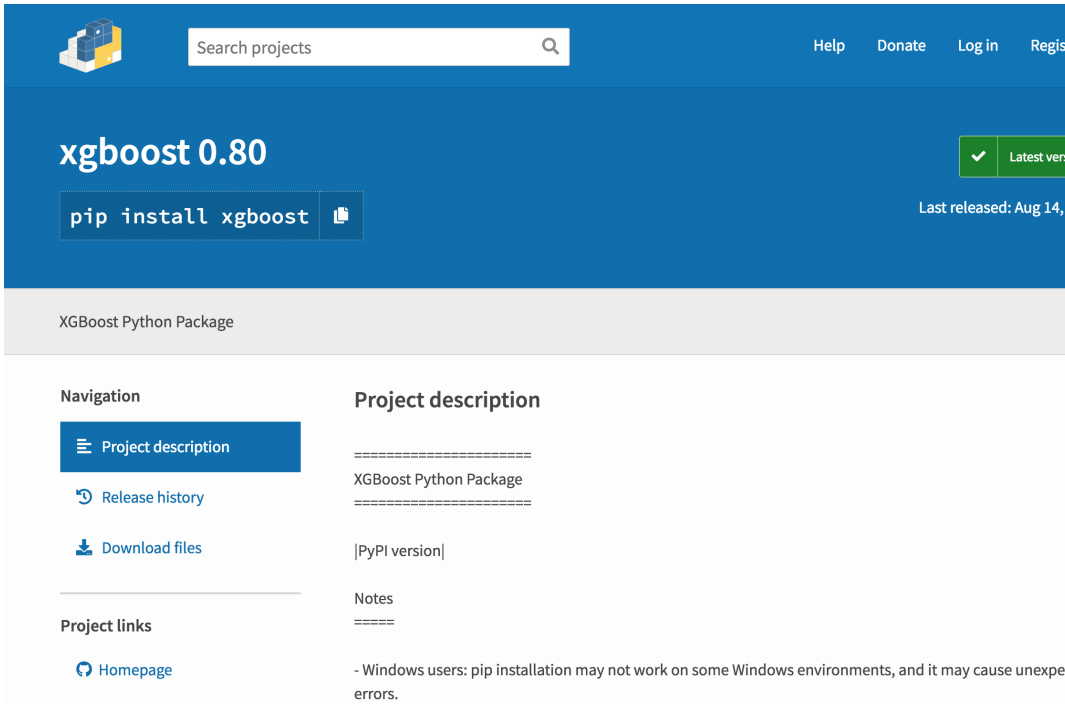
Python modules will follow PEP 427 to provide **wheel**, **tar.gz**, or **zip** as distribution file. So we can follow this specification to find the compatible module and use it with **Vendor**.

Here is an example for download a module **scikit-learn**:

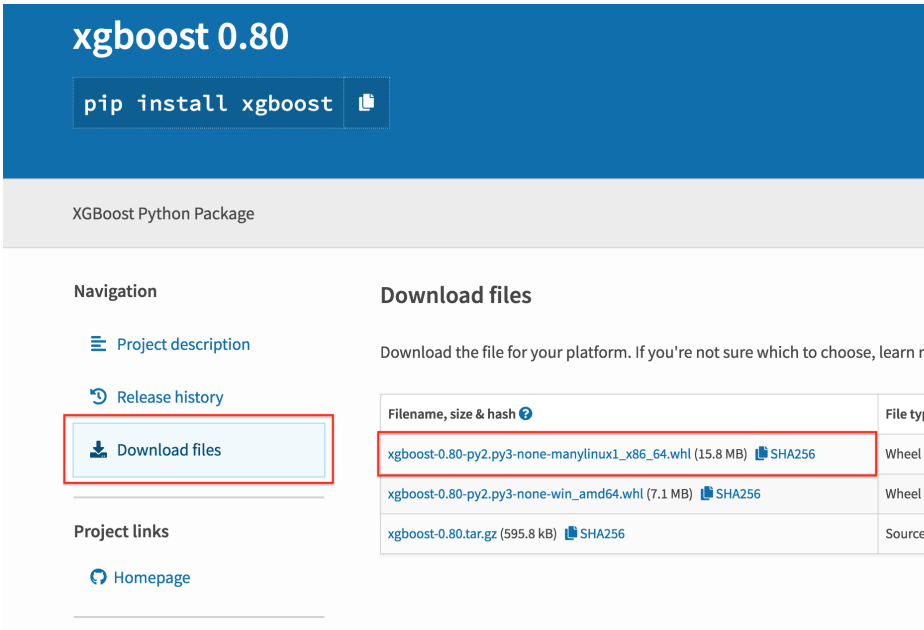
1. Search the module on PyPI.



2. Find the correct project page.



3. Switch to **Download files** page.



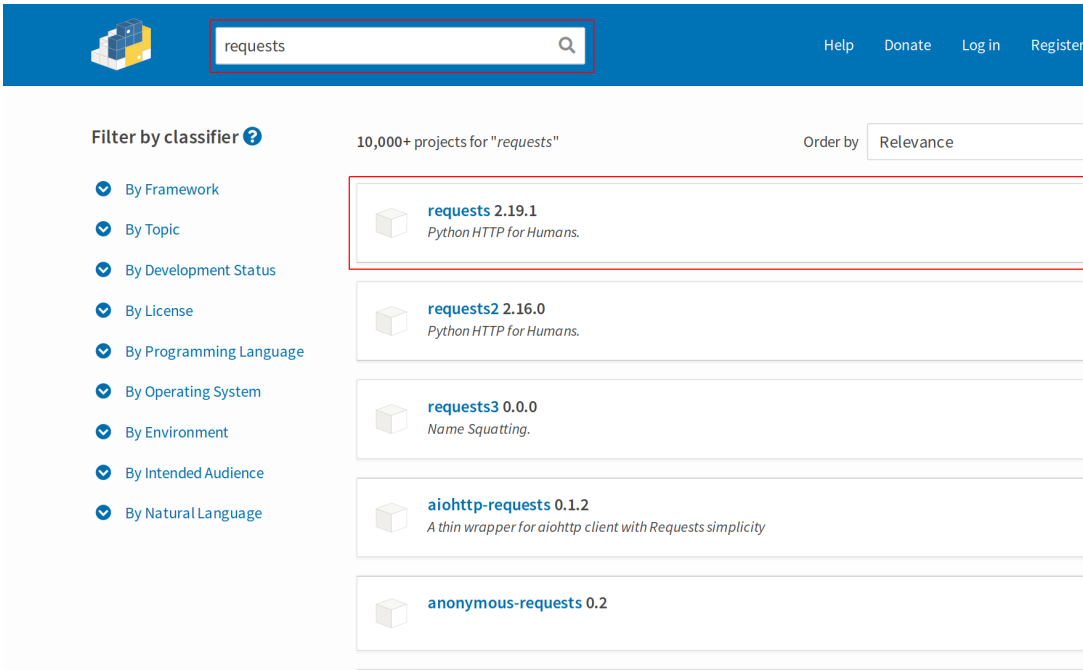
4. Choose compatible version and download.

The name of wheel file looks like:

```
xgboost-0.80-py2.py3-none-manylinux1_x86_64.whl
```

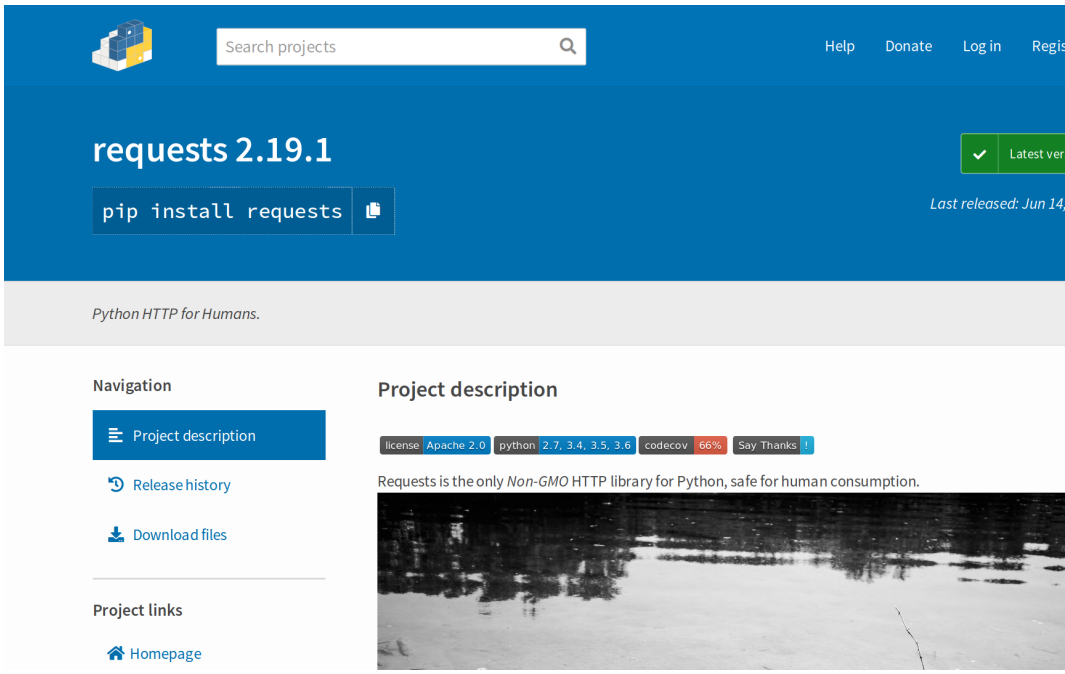
According to PEP 427, **cp36** means it is for Python (more specifically, it's CPython) **3.6.x**, **manylinux1** means it is for **Linux** platform, and **x86_64** means it is for **64bit architecture**.

Another example is **requests**:

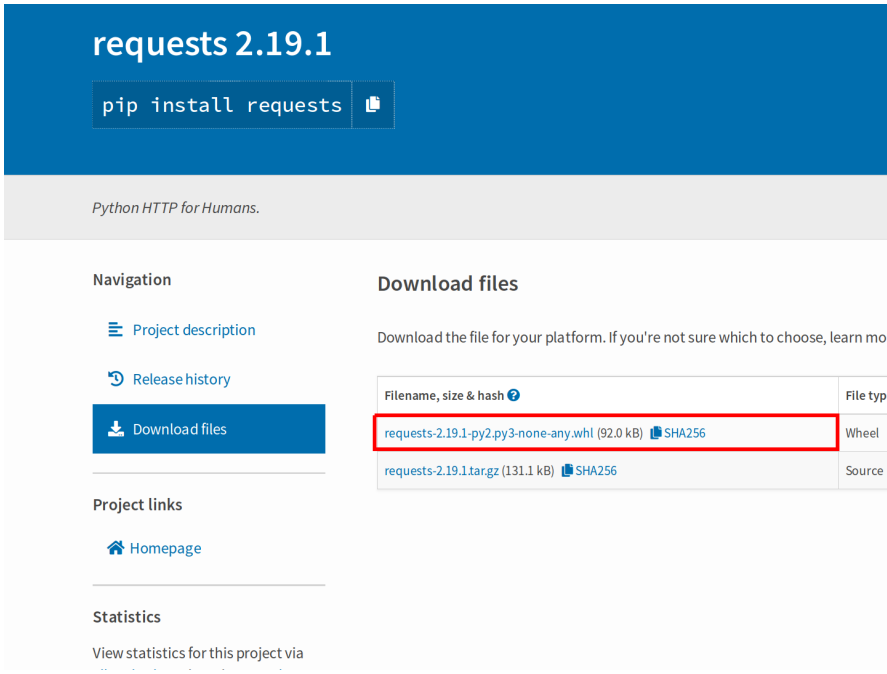


1. Search the module on PyPI.

2. Find the correct project page.



3. Switch to **Download files** page and download.



In this example, the name of wheel is:

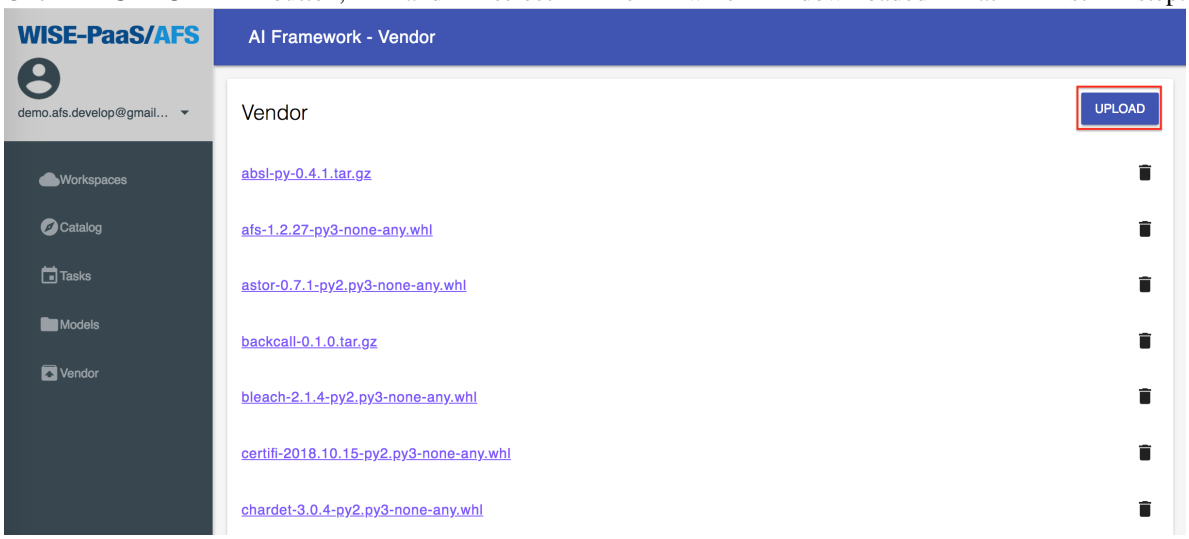
```
requests-2.19.1-py2.py3-none-any.whl
```

If the file name looks like this, it means this wheel can be used for both **Python 2.x** and **Python 3.x** on **any** platform.

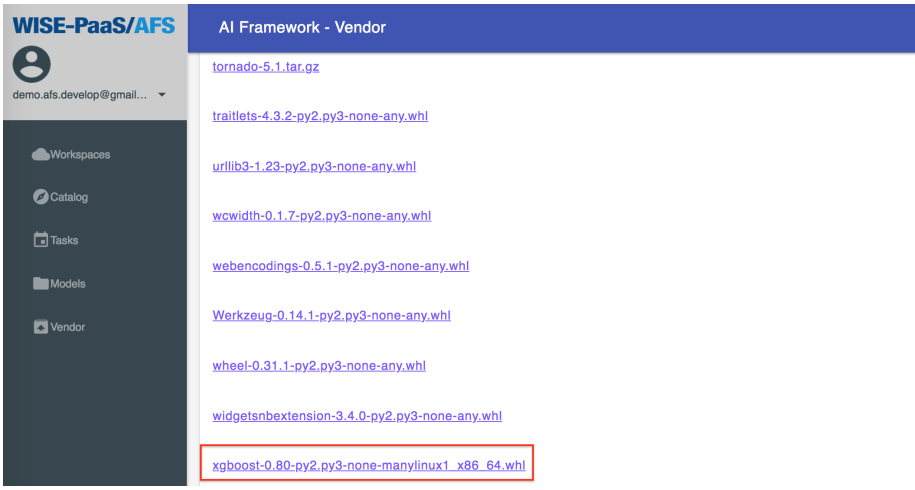
5.2 Upload module to Vendor

After download module file, you can upload it to AFS Vendor and let all analytic app to use this module in Online Code IDE.

1. Click **UPLOAD** button, and select file which downloaded at first step.

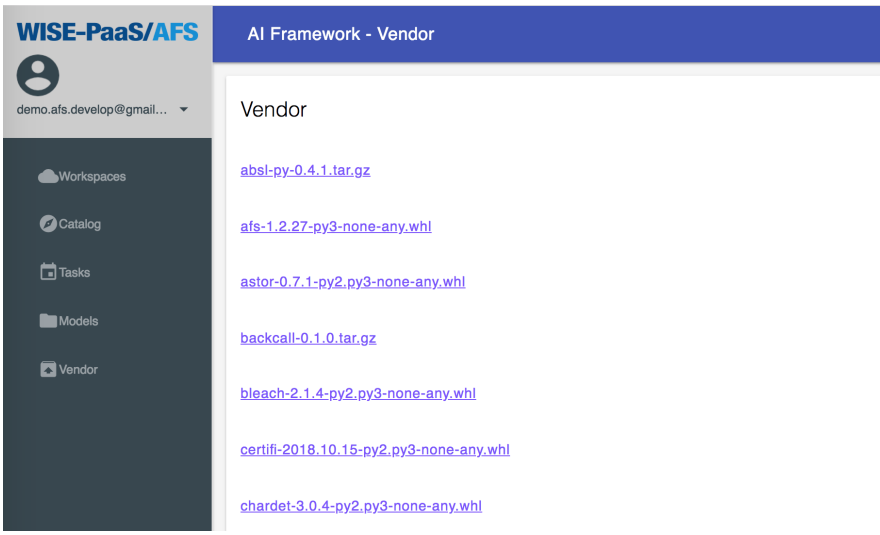


2. After uploading the package, we can find it.

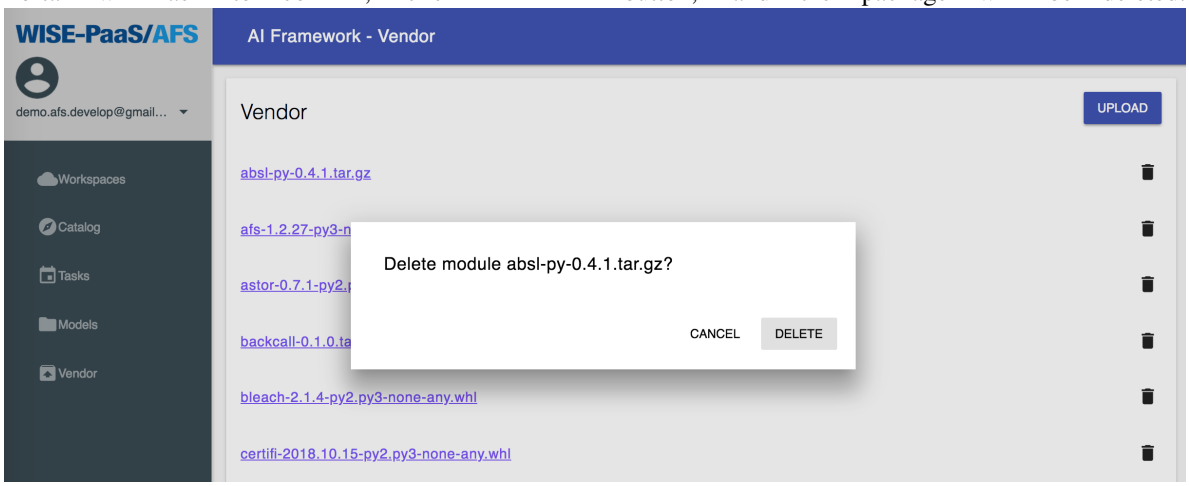


5.3 Delete module in Vendor

You can also delete modules in Vendor with following steps:



1. Click trash icon right behind the module name.
2. Portal will ask to confirm, click **DELETE** button, and the package will be deleted.



CHAPTER 6

AFS SDK

6.1 Documents

Reference documents Readthedocs

6.2 Installation

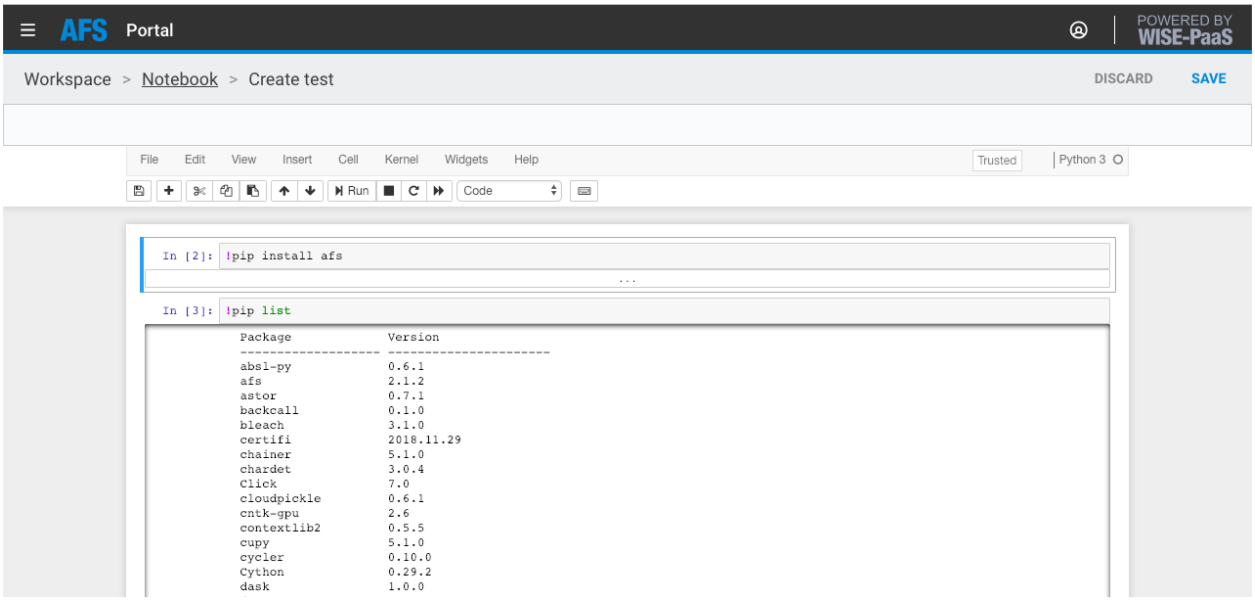
Support python version 3.5 or later

6.2.1 pip install on AFS notebook

AFS provides the release version SDK on private pypi server. Run the following command on notebook cell to install SDK:

```
!pip install afs
```

List the installed packages.



6.3 Develop

6.3.1 (For SDK developer) From sources

1. Clone the repository to local.
2. To build the library run:

```
$ python setup.py install
```

6.3.2 (For SDK developer) Build from source

1. Clone the repository to local.
2. To build the wheel package:

```
$ python setup.py bdist_wheel
```

1. .whl will be in dist/

CHAPTER 7

Install AFS-SDK without external network

If you want install AFS-SDK without external network, you should install dependency step by step. The following is afs-sdk dependency tree:

7.1 How to check out the dependency tree command

```
! pip install pipdeptree
! pipdeptree -fl
```

7.2 How to install module on private cloud

Install module with Vendor in private cloud

7.3 AFS-SDK dependency tree

Install dependency module first.

7.3.1 afs==1.2.28

```
afs
  click
  influxdb
    python-dateuti
      six
    pytz
    requests
      certifi
      chardet
      idna
      urllib3
    six
  pandas
    numpy
    python-dateutil
      six
    pytz
  PyYAML
  requests
    certifi
    chardet
    idna
    urllib3
  urllib3
```

There is a script for installing dependency quickly on AFS online code IDE. And replace the instance_id and workspace_id.

Script

```
import os

# check pkg config, instance id, workspace_id
pkg = ['urllib3-1.23-py2.py3-none-any.whl', 'six-1.11.0-py2.py3-none-any.whl',
↪'python_dateutil-2.7.3-py2.py3-none-any.whl',
 'chardet-3.0.4-py2.py3-none-any.whl', 'certifi-2018.8.24-py2.py3-none-any.whl',
↪'idna-2.7-py2.py3-none-any.whl',
 'click-6.7-py2.py3-none-any.whl', 'requests-2.19.1-py2.py3-none-any.whl', 'influxdb-
↪5.2.0-py2.py3-none-any.whl']
instance_id = '779fd10d-24ee-4603-b18a-dcb279eac8b5'
workspace_id = '0c581c22-e115-4397-b18e-a36a27002762'

install_cmd = '$afs_url/v1/{0}/workspaces/{1}/vendor/'.format(instance_id, workspace_
↪id)
auth_cmd = '?auth_code=$auth_code'

# loop install
for i in pkg:
    cmd = '{0}{1}{2}'.format(install_cmd, i, auth_cmd)
    os.environ['cmd'] = cmd
    !pip install $cmd
```

7.4 (For developer) Build AFS-SDK whl

To build the wheel module:

```
$ python setup.py bdist_wheel
```

AFS-SDK whl file will be in dist/ directory.

CHAPTER 8

Examples

8.1 models

8.1.1 upload_models

How to upload a model file on notebook.

Code

```
from afs import models

# Write a file as model file.
with open('model.h5', 'w') as f:
    f.write('dummy model')

# User-define evaluation result
extra_evaluation = {
    'confusion_matrix_TP': 0.9,
    'confusion_matrix_FP': 0.8,
    'confusion_matrix_TN': 0.7,
    'confusion_matrix_FN': 0.6,
    'AUC': 1.0
}

# User-define Tags
tags = {'machine': 'machine01'}

# Model object
afs_models = models()

# Upload the model to repository and the repository name is the same as file name.
# Accuracy and loss is necessary, but extra_evaluation and tags are optional.
afs_models.upload_model(
    model_path='model.h5', accuracy=0.4, loss=0.3, extra_evaluation=extra_evaluation,␣
↪tags=tags, model_repository_name='model.h5')
```

(continues on next page)

(continued from previous page)

```
# Get the latest model info
model_info = afs_models.get_latest_model_info(model_repository_name='model.h5')

# See the model info
print(model_info)
```

results

```
{
    'evaluation_result': {
        'accuracy': 0.4,
        'loss': 0.3,
        'confusion_matrix': {
            'TP': 0.9,
            'FP': 0.8,
            'TN': 0.7,
            'FN': 0.6
        },
        'AUC': 1.0
    },
    'tags': {
        'machine': 'machine01'
    },
    'created_at': '2018-12-06 08:41:39'
}
```

8.1.2 get_latest_model_info

Code

```
from afs import models
afs_models = models()
afs_models.get_latest_model_info(model_repository_name='model.h5')
```

Output

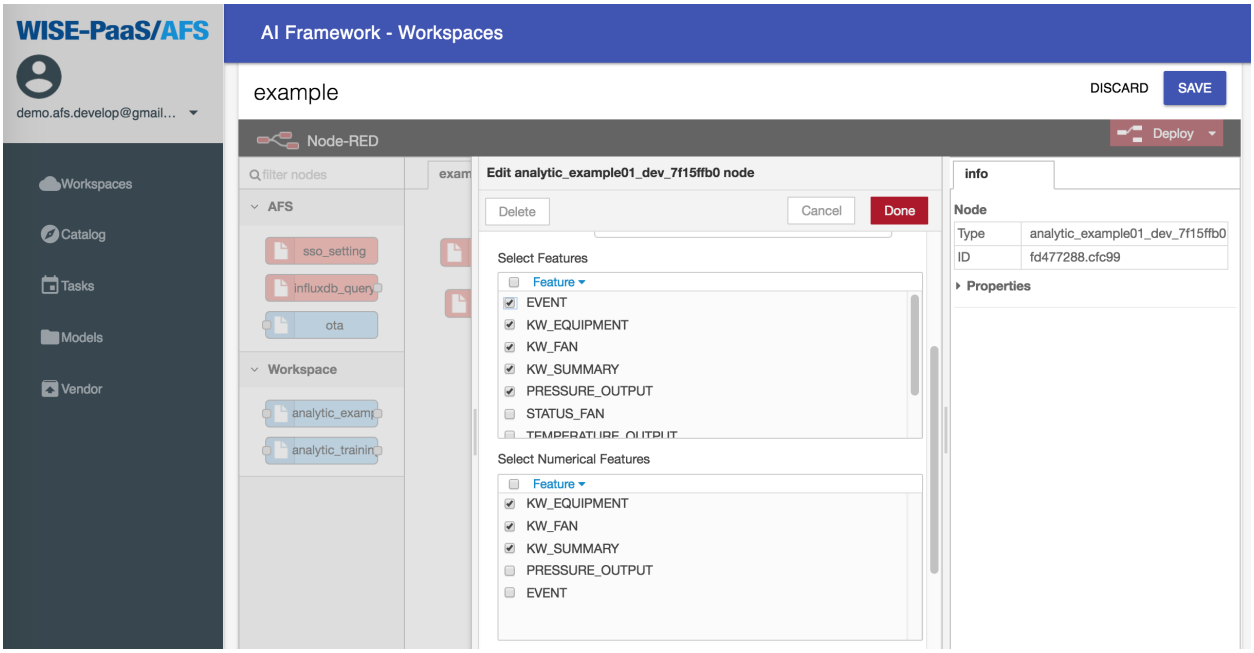
```
{
    'evaluation_result': {
        'accuracy': 0.123,
        'loss': 0.123
    },
    'tags': {},
    'created_at': '2018-09-11 10:15:54'
}
```

8.2 config_handler

8.2.1 Features

How to write a AFS API to get features, including target, select_features, numerical. [Example]

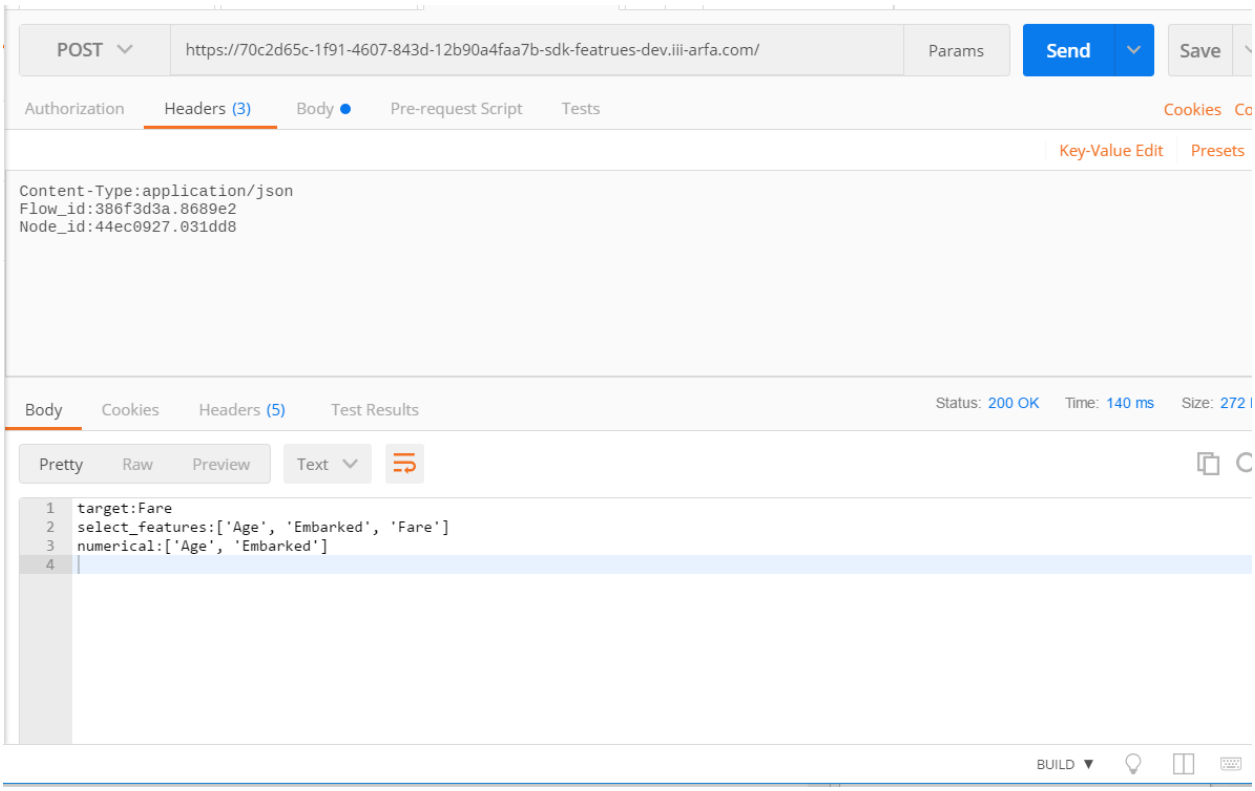
Flow setting



API response

Post Request

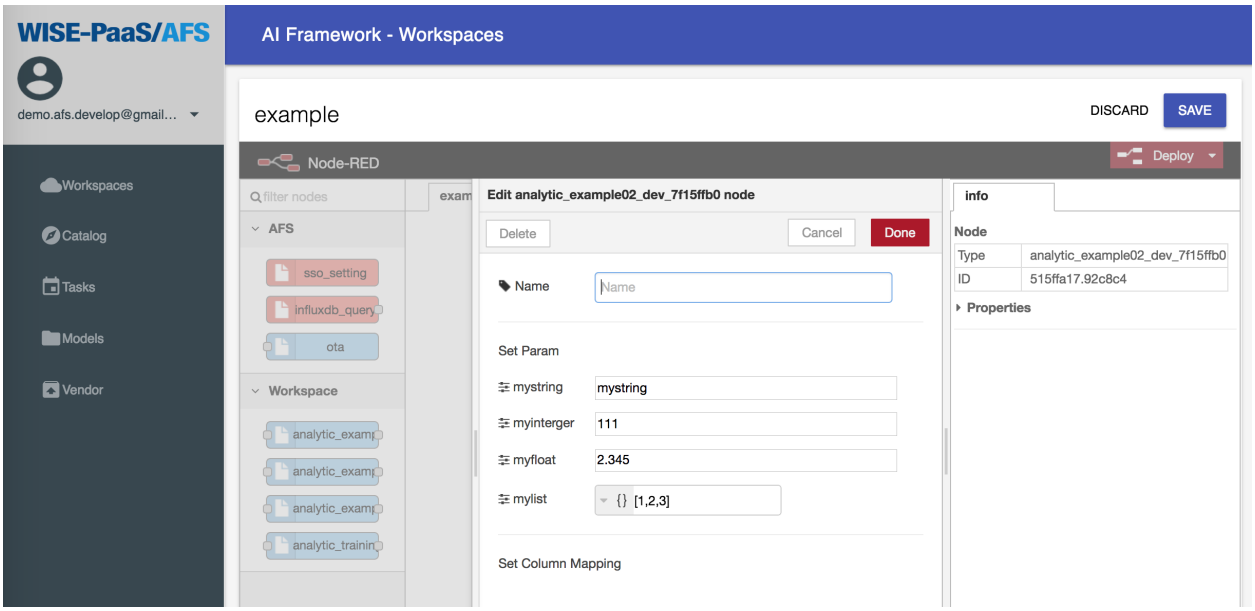
```
{
    "data": {
        "mc": {
            "0": 21
        }
    }
}
```



8.2.2 Parameter (Type string, integet, float, list)

How to write a AFS API to get parameters with types. [Example]

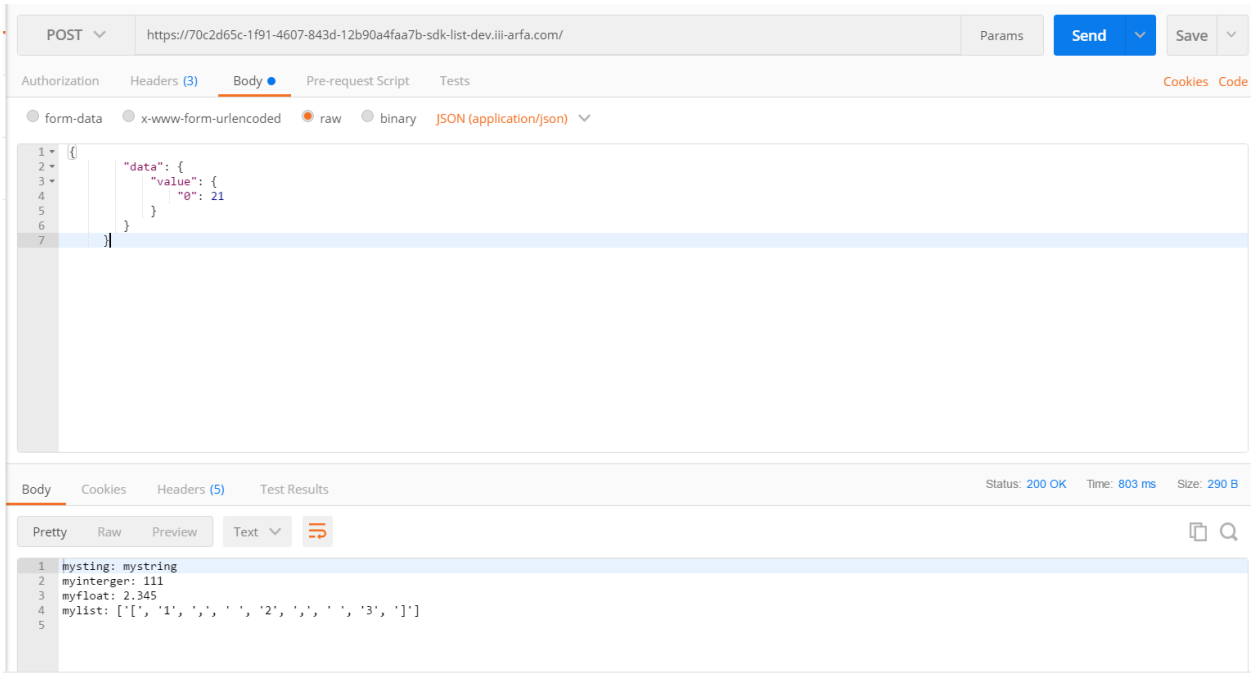
Flow setting



API response

Post Request

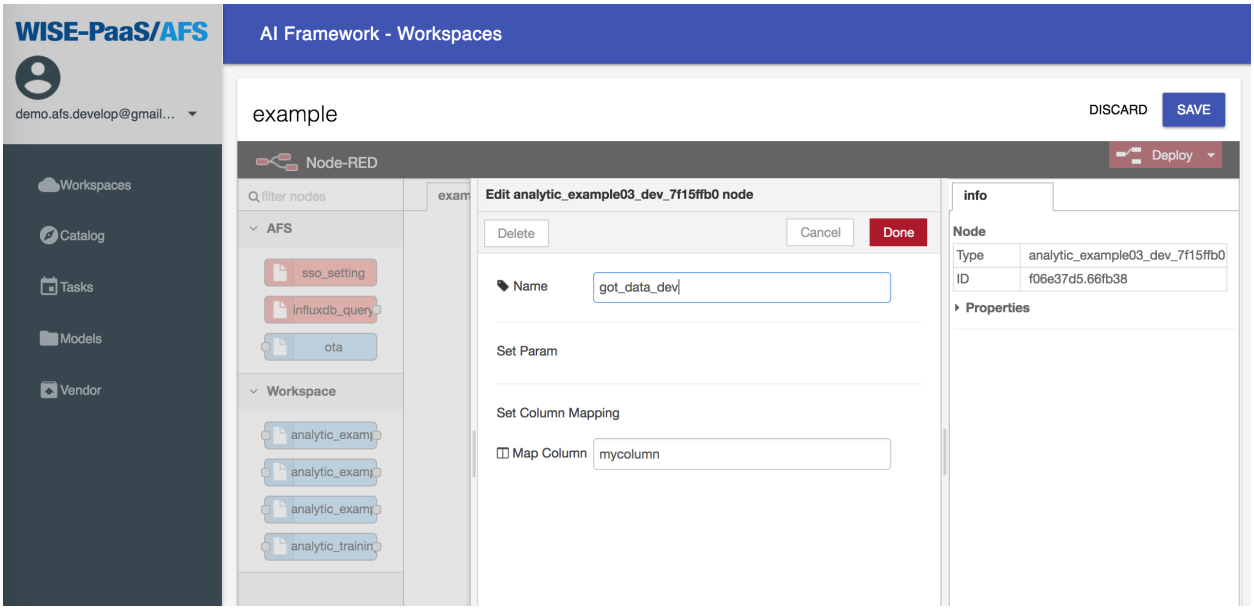
```
{
    "data": {
        "mc": {
            "0": 21
        }
    }
}
```



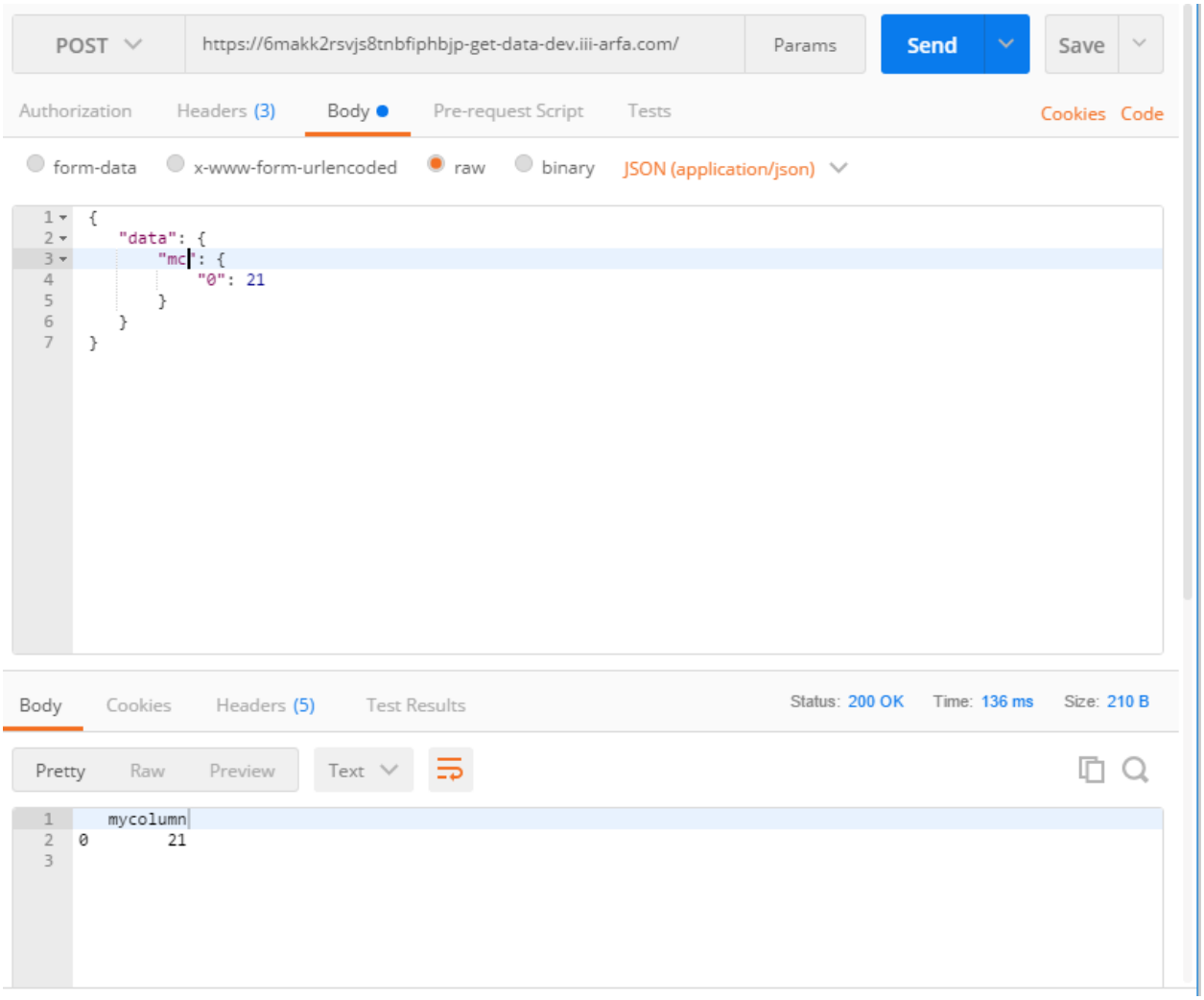
8.2.3 Data

How to write a AFS API to get data. [Example]

Flow setting



API response



8.2.4 API Example using config_handler

Code

```
manifest = {
    'memory': 256,
    'disk_quota': 256,
    'buildpack': 'python_buildpack',
    "requirements":[
        "pandas",
        "afs"
    ],
    'type': 'API'
}
```

```
from afs import config_handler
from pandas import DataFrame
```

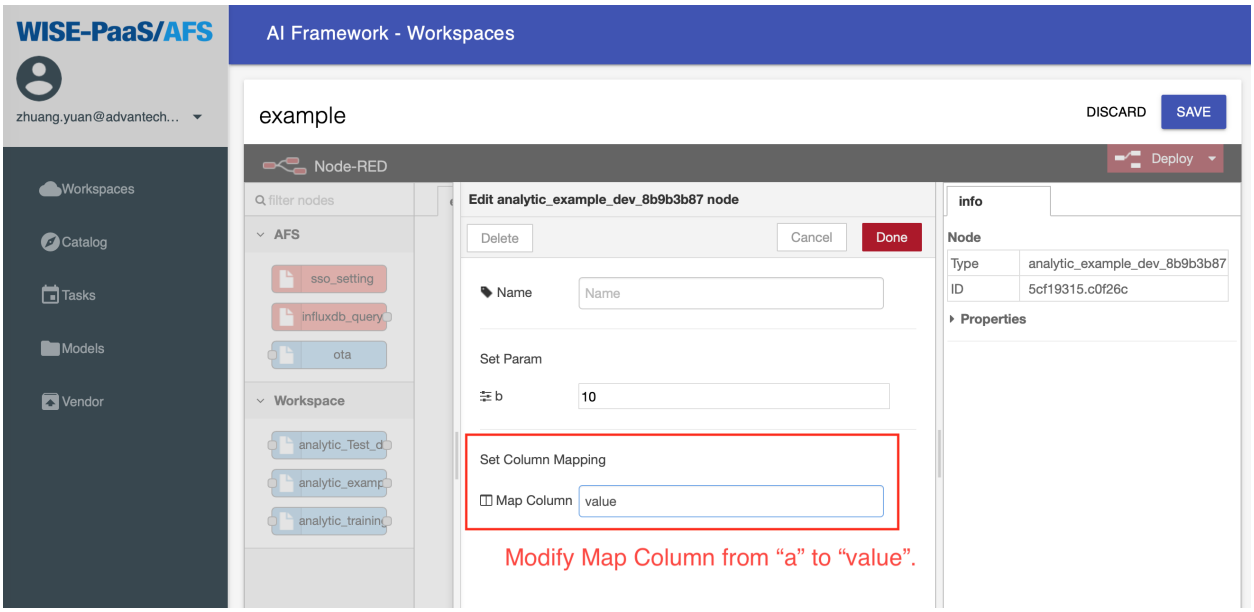
(continues on next page)

(continued from previous page)

```
import json

# Setting API parameters and column name
cfg = config_handler()
cfg.set_param('b', type='integer', required=True, default=10)
cfg.set_column('a')
cfg.summary()
```

```
# POST /

# Set flow architecture, REQUEST is the request including body and headers from client
cfg.set_kernel_gateway(REQUEST)

# Get the parameter from node-red setting
b = cfg.get_param('b')

# Get the data from request, and transform to DataFrame Type
a = cfg.get_data()
result = a + b

# Send the result to next node, and result is  DataFrame Type
ret = cfg.next_node(result, debug=True)

# The printing is the API response.
print(json.dumps(ret))
```

Solution



Request Example

```
{
    "headers": {
        "Flow_id": "b896452e.73d968",
        "Node_id": "fb3d279.613efd8"
    },
```

(continues on next page)

(continued from previous page)

```
    "body": {
        "data": {
            "value": {
                "0": 21
            }
        }
    }
}
```

Response

```
{
    "random": 25,
    "result": {
        "data": {
            "value": {
                "0": 1045
            }
        },
        "node_id": "db4f28d6.59d7e8"
    }
}
```

8.3 Services

How to get the subscribed influxdb credential.

Code

```
from afs import services

myservice =  services()
credential = myservice.get_service_info('influxdb')

# Show one of the credential of the subscribed services.
print(credential)

# Influxdb credential
username = credential['username']
password = credential['password']
host = credential['host']
port = credential['port']
database = credential['database']
```

Output

```
{
    'database': '7cdd5039-59a4-4d78-b911-4ee984183227',
    'password': 'KggwuFtuNQxbxvQQAdJl2WGqw',
    'port': 8086,
    'host': '10.100.20.1',
    'uri': 'http://10.100.20.1:8086',
    'username': 'e821d27d-401e-4db1-8827-20270dfb73e7'
}
```

CHAPTER 9

Command Line Interface

To allow EI-PaaS user push your analytic app from the local machine, **EI-PaaS AFS SDK** provides a **Command Line Interface(CLI)** for users. The CLI only provides one function, to **push** analytic app into your service instance of EI-PaaS AFS.

9.1 Steps

1. Login to AFS with your EI-PaaS SSO user and the target AFS endpoint. For example:

```
eipaas-afs login portal-afs.iii-cflab.com $USERNAME $PASSWORD
```

2. List all service instances for your EI-PaaS SSO user.

```
eipaas-afs service_instances
```

3. Select one of service instance you want to push this analytic app to.

```
eipaas-afs target -s $SERVIE_INSTANCE_ID
```

4. Change your current directory to your analytic app and run the command:

```
eipaas-afs push
```

   This will read the **manifest.yml** and push this analytic app into your workspace. This operation may take a while, just patient.

5. Use AFS portal to check the result.

CHAPTER 10

API Reference

10.1 afs.models module

class afs.models.**models**(*target_endpoint=None*, *instance_id=None*, *auth_code=None*)

Bases: object

create_model_repo(*model_repository_name*)

Create a new model repository. (Support v2 API)

Parameters **repo_name** (*str*) – (optional)The name of model repository.

Returns the new uuid of the repository

delete_model(*model_name*, *model_repository_name=None*)

Delete model.

Parameters

- **model_name** – model name.
- **model_repository_name** – model repository name.

Returns bool

delete_model_repository(*model_repository_name*)

Delete model repository.

Parameters **model_repository_name** – model repository name.

Returns bool

download_model(*save_path*, *model_repository_name=None*, *model_name=None*, *last_one=False*)

Download model from model repository to a file.

Parameters

- **model_repository_name** (*str*) – The model name exists in model repository
- **save_path** (*str*) – The path exist in file system

get_latest_model_info(*model_repository_name=None*)
Get the latest model info, including created_at, tags, evaluation_result. (Support v2 API)

Parameters **model_repository_name** – (optional)The name of model repository.

Returns dict. the latest of model info in model repository.

get_model_id(*model_name=None*, *model_repository_name=None*, *last_one=True*)
Get model id by model name.

Parameters

- **model_name** (*str*) – model name. No need if last_one is true.
- **model_repository_name** (*str*) – model respository name where the model is.
- **last_one** (*bool*) – auto get the model_repository last one model

Returns str model id

get_model_info(*model_name*, *model_repository_name=None*)
Get model info, including created_at, tags, evaluation_result. (V2 API)

Parameters

- **model_name** – model name
- **model_repository_name** – The name of model repository.

Returns dict model info

get_model_repo_id(*model_repository_name=None*)
Get model repository by name.

Parameters **model_repository_name** (*str*) –

Returns str model repository id

switch_repo(*model_repository_name=None*)
Switch current repository. If the model is not exist, return none. (Support v2 API)

Parameters **repo_name** (*str*) – (optional)The name of model repository.

Returns None, repo_id, exception

upload_model(*model_path*, *accuracy=None*, *loss=None*, *tags={}*, *extra_evaluation={}*, *model_repository_name=None*, *model_name=None*)
Upload model_name to model repository.If model_name is not exists in the repository, this function will create one.(Support v2 API)

Parameters

- **model_path** (*str*) – (required) model filepath
- **accuracy** (*float*) – (optional) model accuracy value, between 0-1
- **loss** (*float*) – (optional) model loss value
- **tags** (*dict*) – (optional) tag from model
- **extra_evaluation** (*dict*) – (optional) other evaluation from model
- **model_name** (*str*) – (optional) Give model a name or default auto a uuid4 name

Returns bool

10.2 afs.services module

class afs.services.**services**(*target_endpoint=None*, *instance_id=None*, *auth_code=None*)
Bases: object

get_service_info(*service_name*, *service_key=None*)
Get the subscribed service one of key.

Parameters

- **service_name** (*str*) – (required) the service on EI-PaaS was subscribed
- **service_key** (*str*) – (optional) specific service key. Default is None, pick one of keys.

get_service_list()
List all credentials which the services you subscribed.

Returns list. credential info

10.3 afs.config_handler module

class afs.config_handler.**config_handler**
Bases: object

get_column()
Get the column mapping list.

Returns The value is the column name would use in the AFS API, and the key is the mapping column name.

Return type dict

get_data()
Transform REQUEST data to DataFrame type.

Returns DataFrame type. Data from REQUEST and rename column name.

get_features_numerical()
Get feature numerical from flow json.

Returns feature numerical list

Return type list

get_features_selected()
Get feature selected from flow json.

Returns feature select list

Return type list

get_features_target()
Get feature target from flow json.

Returns feature target name

Return type str

get_param(*key*)
Get parameter from the key name, and it should be set from set_param.

Parameters **key** (*str*) – The parameter key set from method set_param

Returns Specfic type depends on set_param. The value of the key name.

next_node(*data*, *debug=False*)
Send data to next node according to flow.

Parameters

- **data** – DataFrame type. Data will be sent to next node.
- **debug** (*bool*) – If debug is True, method will return response message from the next node.

Returns Response JSON

Return type dict

set_column(*column_name*)
The column name will be used in the AFS API.

Parameters **column_name** (*str*) – The column name used in the following API

set_features(*enable=False*)
The feature name will be used in the AFS API.

Parameters **feature_list** (*list*) – The feature name used in the following API

set_kernel_gateway(*REQUEST*, *flow_json_file=None*, *env_obj={}*)
For Jupyter kernel gateway API, REQUEST is the request given by kernel gateway. Reference REQUEST: http://jupyter-kernel-gateway.readthedocs.io/en/latest/http-mode.html

Parameters

- **REQUEST** (*str*) – Jupyter kernel gateway request.
- **env_obj** (*dict*) – Key names are VCAP_APPLICATION, afs_host_url, node_host_url, afs_auth_code, sso_host_url, rmm_host_url(option).
- **flow_json_file** (*str*) – String of file path. For debug, developer can use file which contains the flow json as the flow json gotten from NodeRed.

set_param(*key*, *type='string'*, *required=False*, *default=None*)
Set API parameter will be used in the AFS API.

Parameters

- **key** (*str*) – The key name for this parameter
- **type** (*str*) – The type of the paramter, including integer, string or float.
- **required** (*bool*) – The parameter is required or not
- **default** (*str*) – The parameter is given in default

summary()
Summary what parameters and column the AFS API need.This method should be called by the last line in the 2nd cell.

10.4 afs.flow module

class afs.flow.**flow**(*mode='node'*, *env_obj={}*)
Bases: object

exe_next_node(*data={}*, *next_list=None*, *debug=False*)
Request next node api to execute. Dependency: get_node_item(), set_headers()

Parameters

- **next_list** – (list) list of next nodes.
- **data** – (dict) data will send to next node. (dataframe dict)
- **debug** – (bool) whether for debug use. (default=False)

Return error_node (string) node id with error occur.

get_afs_credentials(*sso_token*)
Get AFS credentials about service name, service key.

Parameters **sso_token** – (string) sso token

Return resp (string) response afs credentials list

Return status (int) status code

get_firehose_node_id()
Find node id of firehose type in flow. (check for key name: _node_type)

Return node_id (string) node id of firehose if do not find node_id, function will return ''.

get_flow_list()
Call Node-RED api to get flow list.

needed variable: flow_id, node_host_url

generate: flow_list (list) all nodes in this flow_id. if not exist, variable will be None.

Return flow_list (list) flow list from Node-RED (if can not get flow list from Node-RED api, throw exception.)

get_flow_list_ab(*result*)

get_node_item(*select_node_id*, *is_current_node=True*)
Get Node-RED item from flow_list.

Parameters

- **select_node_id** – (string) node id in Node-RED, for select node.
- **is_current_node** – (bool) This node id is current node. True: Set this node information into node_obj. False: Do not set this node information into node_obj.

Return node (dict) get this node setting information. if not exist, throw exception.

get_sso_node_id()
Find node id of sso_setting type in flow. (check for key-value: type=sso_setting)

Return node_id (string) node id of sso if do not find node_id, function will return ''.

get_sso_token(*req_body*)
Get SSO token.

Parameters **req_body** – (dict) request body for request sso api. {username, password}

Return resp (string) response sso token

Return status (int) status code

set_flow_config(*obj*)
Set config(class properties value) of flow.

Parameters **obj** – (dict) request headers. {flow_id, node_id}

Return is_success (bool) flow config information is setting success. True: setting success. False: lose config information.

set_headers()
Generate headers object for request headers.

Return obj (dict) request headers object. {Content-Type, flow_id, node_id}

10.5 afs.GetJointTable module

class afs.get_joint_table.**GetJointTable**
Bases: object

Parameters

- **query_date** (*dict*) – DATE_FROM from date require to joint, format: %YYYY-%MM-%DD. DATE_TO: to date require to joint, format: %YYYY-%MM-%DD.
- **grafana_dict** (*dict*) – GRAFANA_HOST Grafana endpoint, for example: http://grafana.wise-paas.com/. GRAFANA_USERNAME: Username of Grafana, require premission to get annotation. GRAFANA_PASSWORD: Password of Grafana user. GRAFANA_TAG1: First tag require to merge. GRAFANA_TAG2: Second tag require to merge.
- **idb_dict** (*dict*) – IDB_HOST InflixDB endpoint, for example: http://inflixdb.wise-paas.com. IDB_PORT: Port of InfluxDB. IDB_DBNAME: InfliuxDB database. IDB_USERNAME: Username of InfluxDB, require premission to read. IDB_PASSWORD: Passoed of InfluxDB user
- **tag** (*str*) – tag name which require to merge

10.6 afs.parsers module

afs.parsers.**config_to_dict**(*source*, *startswith='node_config'*)
Transform config(manifest or node_config) from jupyter source code to python dict.

Parameters **source** (*str*) – config source code in jupyter.

Return config transform config from source code to dictionary.

Rtypr dict

afs.parsers.**manifest_parser**(*notebook_path*, *pypi_endpoint*, *output_dir=None*, *manifest_yaml=False*, *afs_sdk_version=None*)
The method parses the manifest in notebook, including manifest.json, requirements.txt, runtime.txt, startup.sh.

Parameters

- **notebook_path** (*str*) – the path of notebook (.ipynb) will be parsed.
- **pypi_endpoint** (*str*) – the requirement would be specific pypi server
- **output_dir** (*str*) – the files would be output in specific path. Default is current directory
- **manifest_yaml** (*bool*) – write manifest.yml or not
- **afs_sdk_version** (*str*) – parse manifest to specific afs sdk version requirement

Returns True or raise exception

CHAPTER 11

Indices and tables

- genindex
- modindex
- search

CHAPTER 12

Inference Engine Install Python Package

The **Inference Engine** is a Python runtime program that runs on Docker in a foggy device, so it is sometimes necessary to update the relevant suites that Python needs.

12.1 Update the Python Package via the Internet

Example below is the installation of the xgboost Package:

1. Enter the container.

```
docker exec -it $CONTAINER_ID bash
```

2. Execute pip install.

```
pip install xgboost
```

```
root@ee698a856aae:/# pip install xgboost
Collecting xgboost
  Downloading https://files.pythonhosted.org/packages/06/7a/442f7da21792566012e5c7e5a7dffa44c1b6cc05c0
8/xgboost-0.72.1-py2.py3-none-manylinux1_x86_64.whl (18.4MB)
    100% |████████████████████████████████| 18.4MB 1.6MB/s
Requirement already satisfied: scipy in /usr/local/lib/python3.6/site-packages (from xgboost) (1.1.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/site-packages (from xgboost) (1.14.4)
Installing collected packages: xgboost
Successfully installed xgboost-0.72.1
You are using pip version 10.0.1, however version 18.0 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

12.2 Update the Python Package via the whl file in a On-Premises environment

Example below is the installation of the xgboost Package:

1. Put the xgboost package whl file in the c:\inference_engine directory.

2. Go to the /inference_engine folder in the container that Docker runs.

```
cd /inference_engine/
```

3. Use pip install to install xgboost's whl file.

```
pip install xgboost-0.72.1-py2.py3-none-manylinux1_x86_64.whl
```

```
root@0cb25a5d7b72:/inference_engine# pip install xgboost-0.72.1-py2.py3-none-manylinux1_x86_64.whl
Processing ./xgboost-0.72.1-py2.py3-none-manylinux1_x86_64.whl
Requirement already satisfied: scipy in /usr/local/lib/python3.6/site-packages (from xgboost==0.72.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/site-packages (from xgboost==0.72.1)
Installing collected packages: xgboost
Successfully installed xgboost-0.72.1
You are using pip version 10.0.1, however version 18.0 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

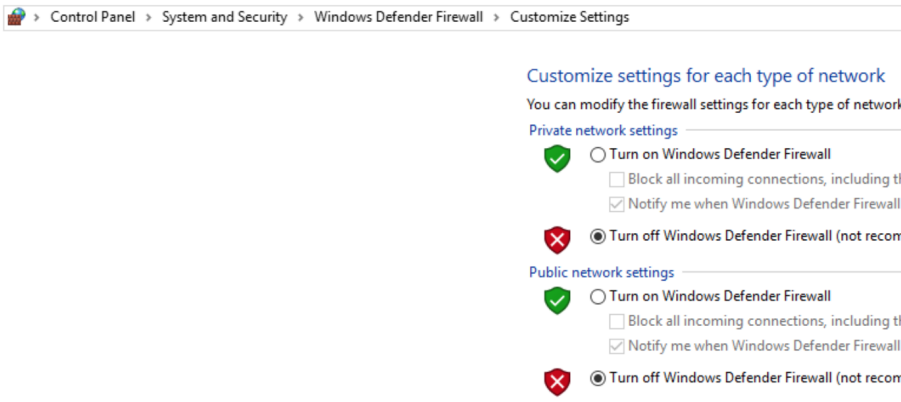
CHAPTER 13

Inference Engine Install Automatically in Edge Device

Previously, an introduction of **Inference Engine**, it's a Python runtime program on Docker. We can install it manually step by step. However, for the industial application, there are many edge devices (e.g., perhaps 100, 1000, or more devices) work online at the same time. In the section, we introduce how to install the Inference Engine automatically in many edge devices.

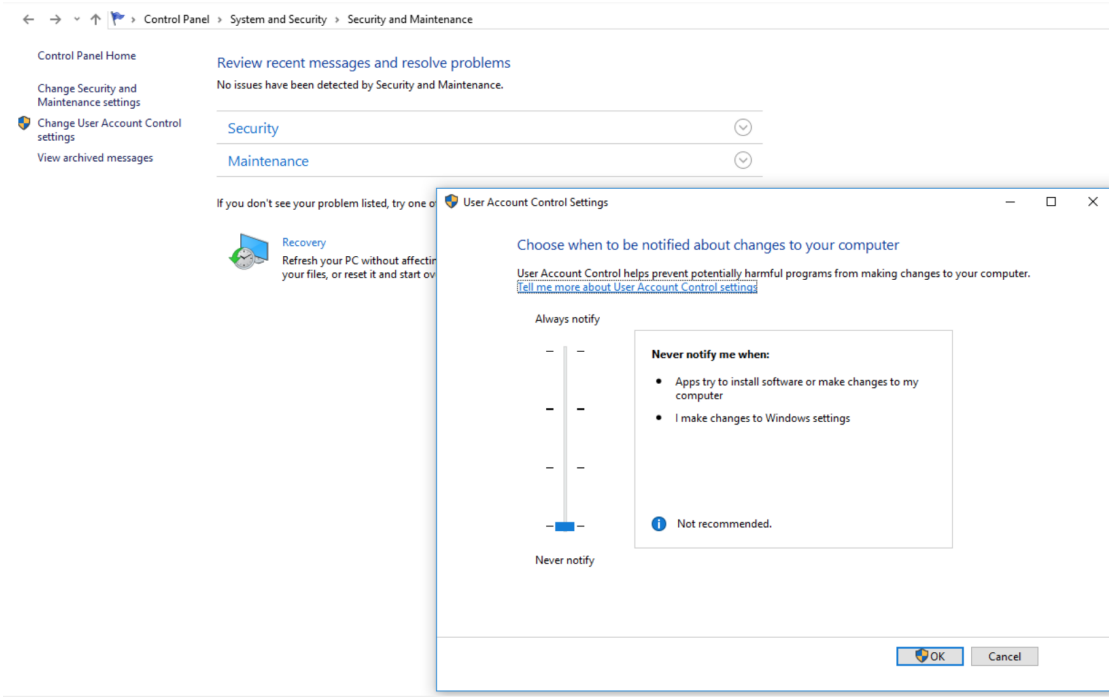
13.1 Pre-condition

- The OS of edge devices must be the **Windows 10 Pro** 64-bit version, and **Build 14393 or later**.
- The language of OS must be in **Simplified Chinese**, **Traditional Chinese**, and **English**.
- Turn on the Hyper-V in Windows 10. About the steps, please refer the document.
- The edge devices must be installed the **RMM Agent (v-1.0.16)**, and registed in RMM Server.
- Get the application of packaging (OTAPackager-1.0.5.exe). [Download]
- Download the files for package as follows:
  - Docker installer. [Download]
  - Three .bat files (include install_docker.bat, start_docker.bat, start_inference.bat). [Download]
  - SSL credential (registry.cert). [Download]
- Setup for login automatically after rebooting, please refer the page.
- Close the firewall.
  - Control Panel > System and Security > Windows Defender FireWall > Customize Settings.



  - Turn off Windows Defender Firewall.

- Close the notification.
  - Control Panel > System and Security > Security and Maintenance > Change User Account Control settings.



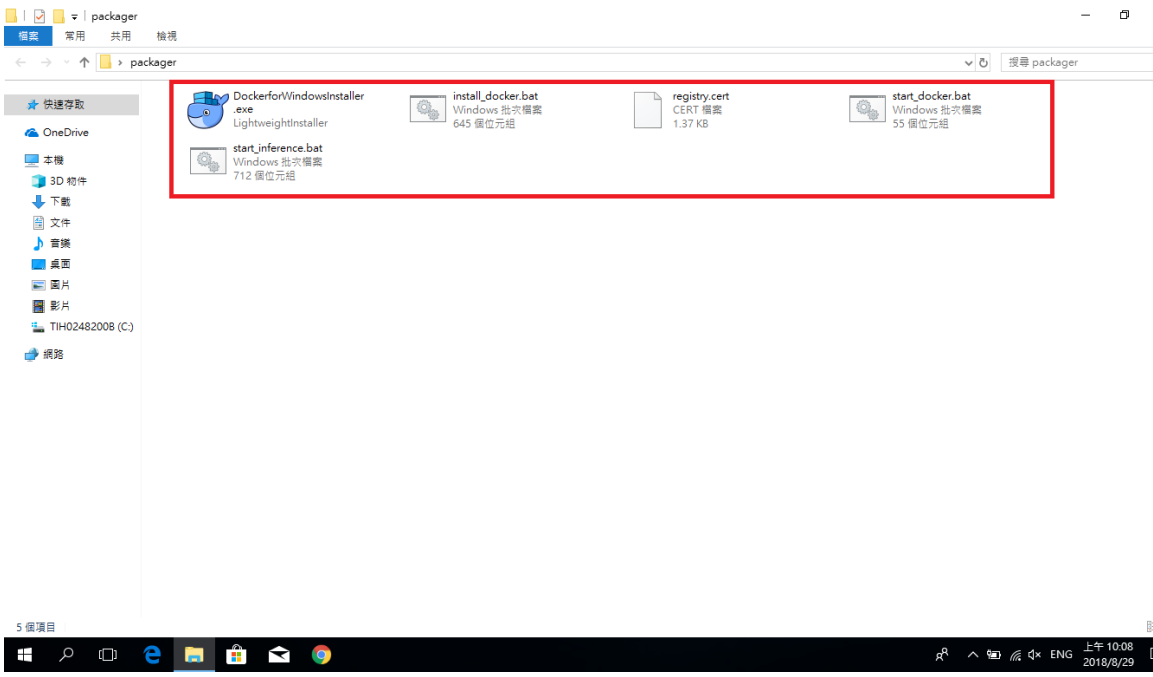
  - Set “Never notify”.

- The docker offical suggestion before installing, please refer the docker docs.
  - Windows 10 64bit: Pro, Enterprise or Education (1607 Anniversary Update, Build 14393 or later).
  - Virtualization is enabled in BIOS. Typically, virtualization is enabled by default. This is different from having Hyper-V enabled. For more detail see Virtualization must be enabled in Troubleshooting.
  - CPU SLAT-capable feature.
  - At least 4GB of RAM.

13.2 Start to Install Inference Engine

1. Use the OTApackager APP to package the required files.

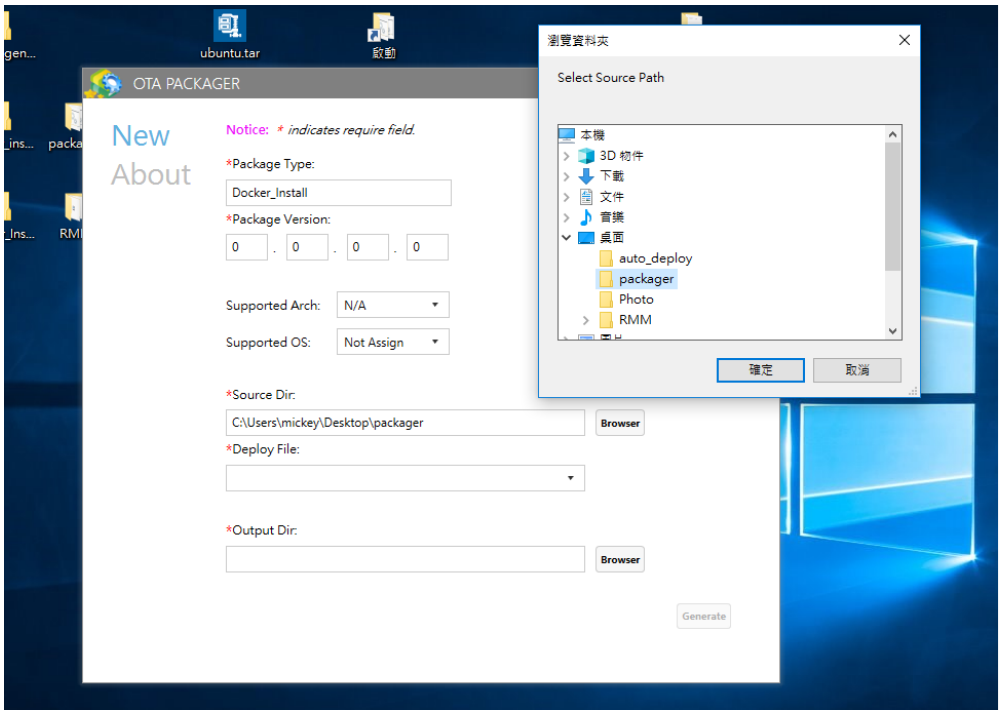
a. The required files.



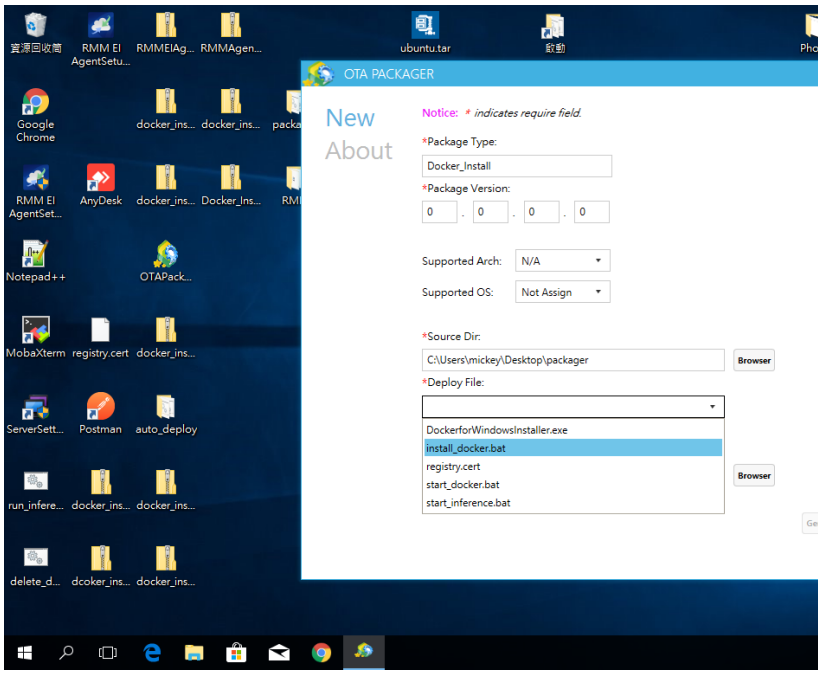
b. Edit "install_docker.bat", the file path should be modified to matching the path in the edge device.

```
copy /Y start_docker.bat "C:\Users\kai\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\start_docker.bat"
copy /Y start_inference.bat "C:\Users\kai\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\start_inference.bat"
certutil -addstore "TrustedPublisher" registry.cert
set file="C:\Program Files\Docker\Docker\Docker for Windows.exe"
if exist %file% (
  echo file is exists
)else (
  "Docker for Windows Installer.exe" install --quiet -Verb RunAs
   net localgroup docker-users kai /add
)
set docker_deamon="C:\Users\kai\.docker"
if not exist %docker_deamon% (
  md C:\Users\kai\.docker
)
(
echo { "registry-mirrors": [], "insecure-registries": [ "23.98.43.195:443" ], "debug": true, "experimental": false}
)>"C:\Users\kai\.docker\daemon.json"
DISM /Online /Enable-Feature /All /FeatureName:Microsoft-Hyper-V /Quiet
shutdown.exe /r /t 90
```

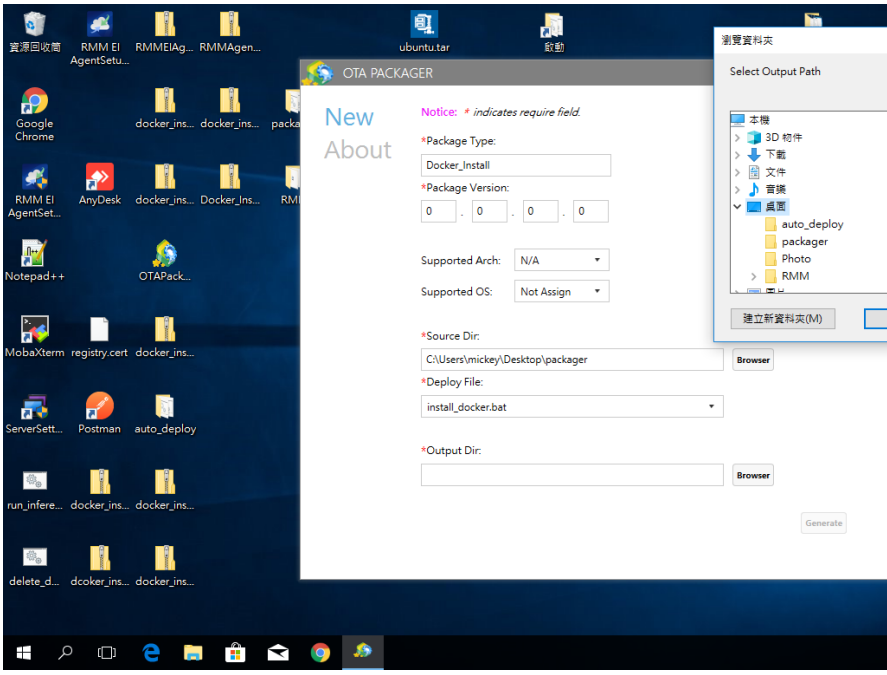
c. Enter the Package Tyep, Package Version, then select the path for saving the package file.



d. Select **install_docker.bat** to be the "Deploy File".

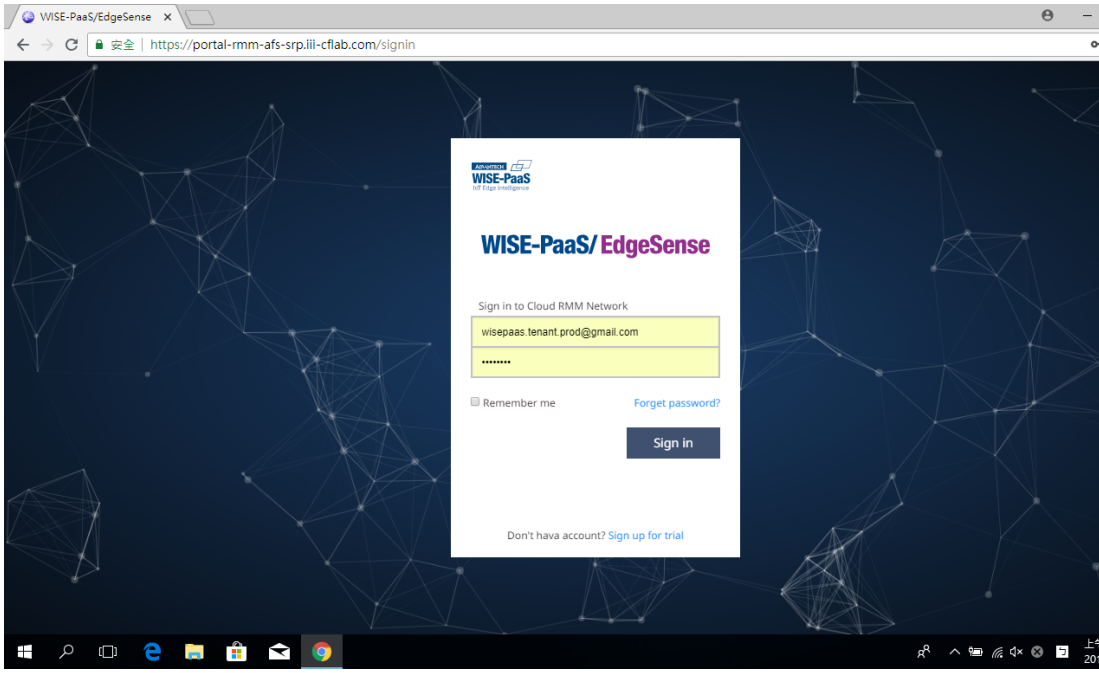


e. Select the folder for saving the package file.

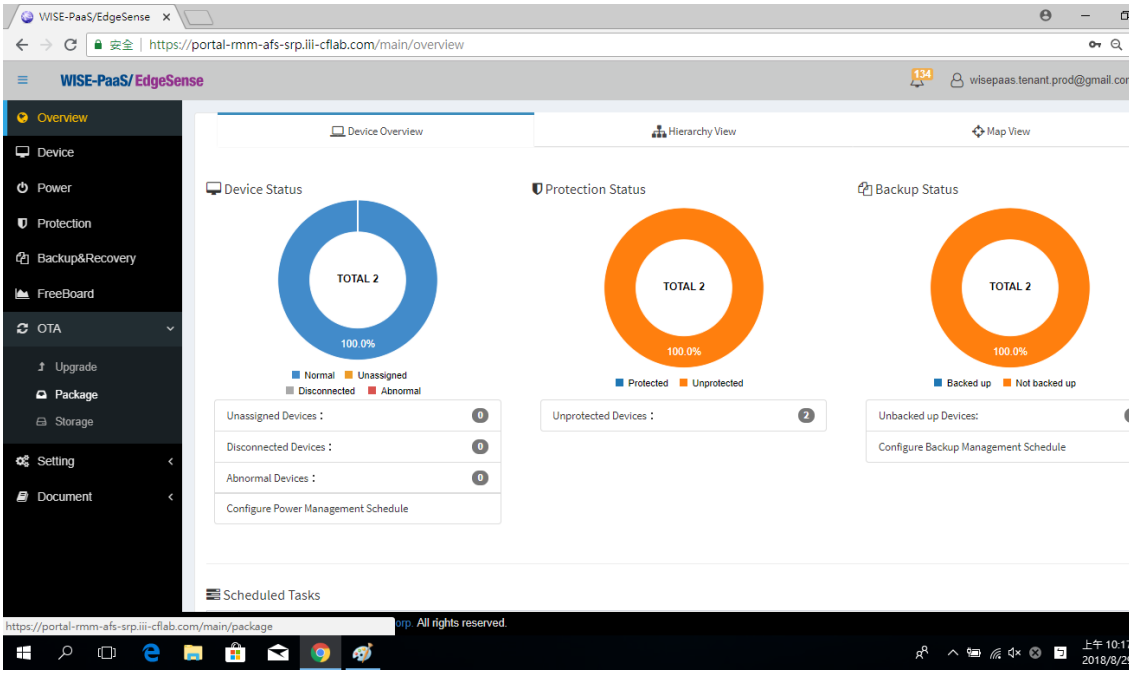


2. Login to **RMM Portal**, and upload the package file.

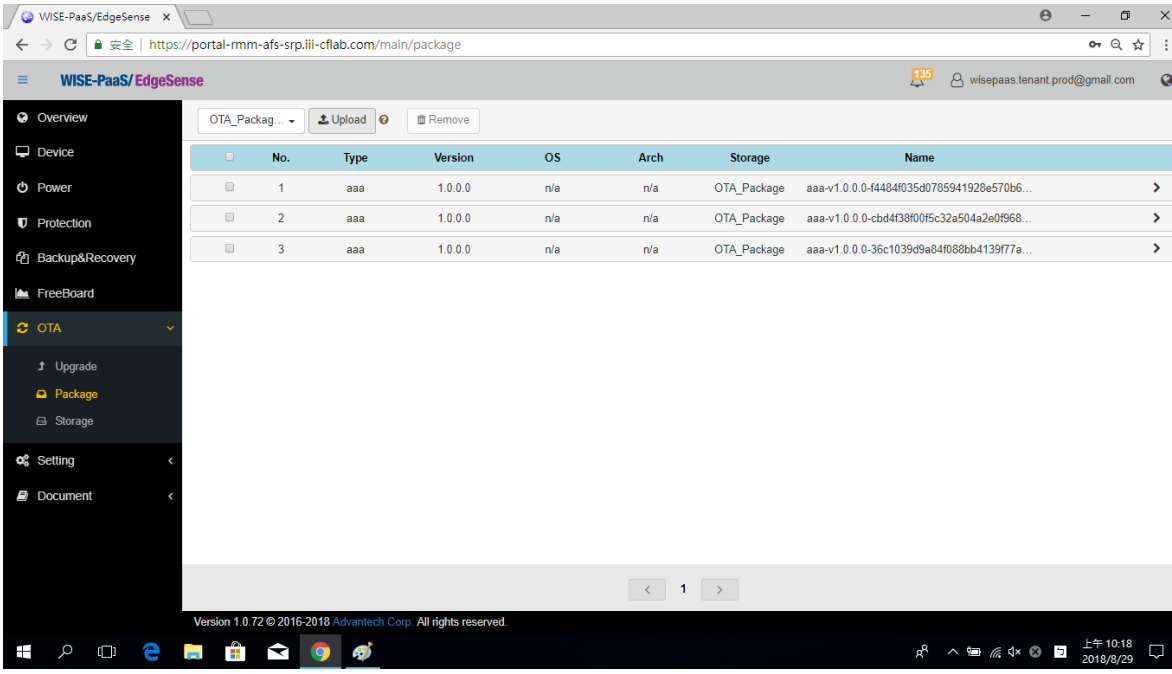
a. Login to **RMM Portal**.



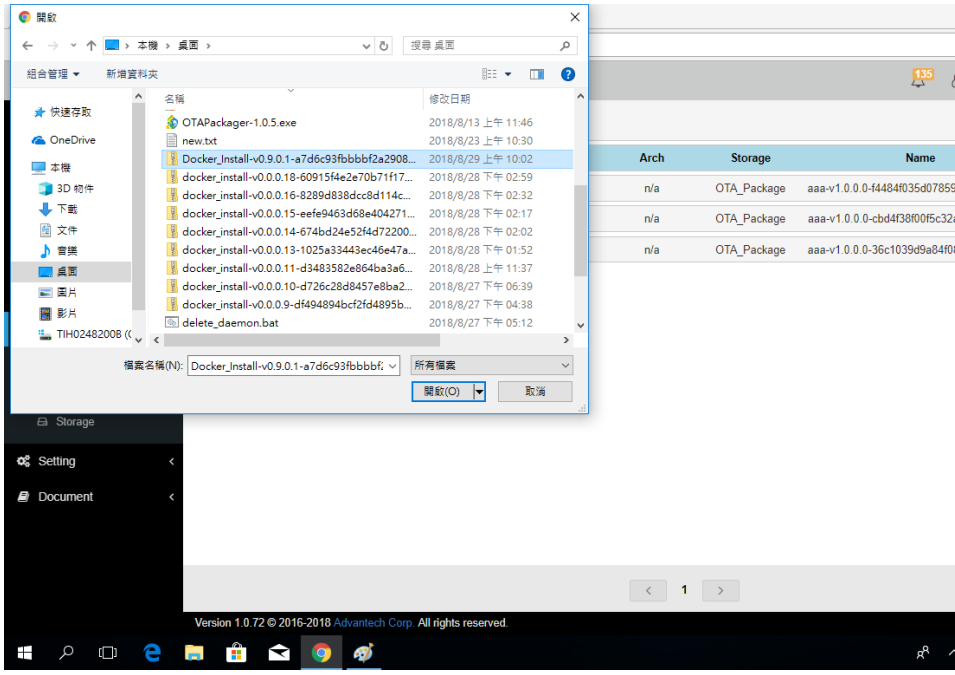
b. Click OTA Package.



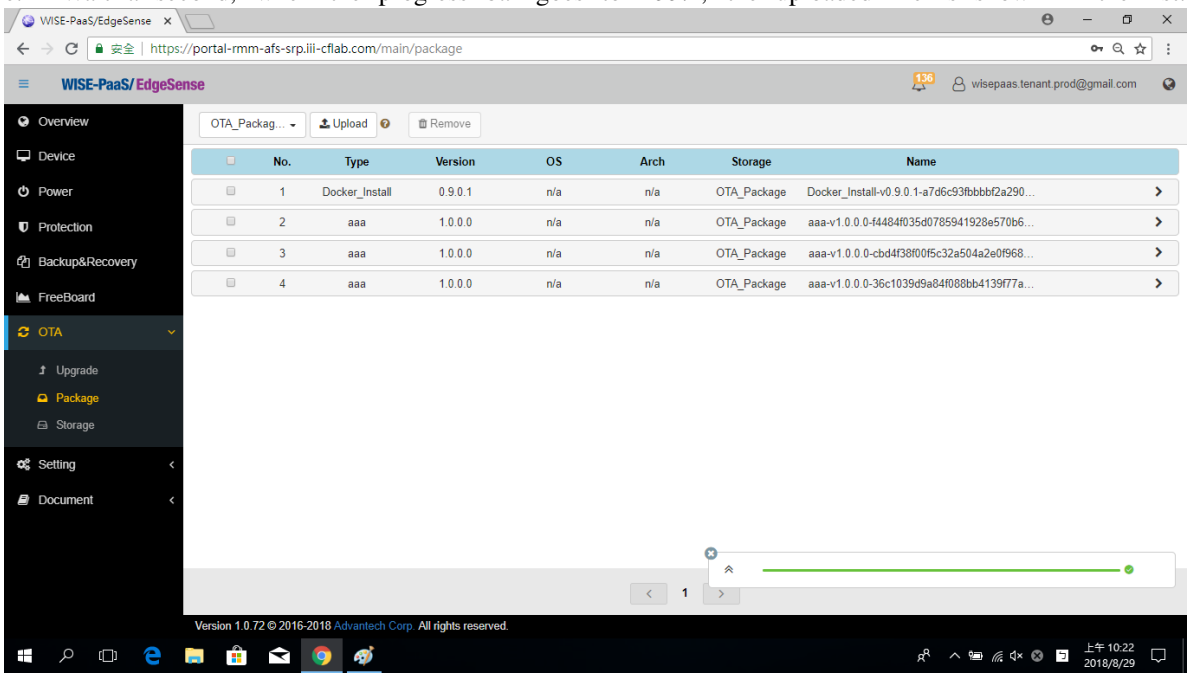
c. Click “Upload”.



d. Select the package file for uploading.

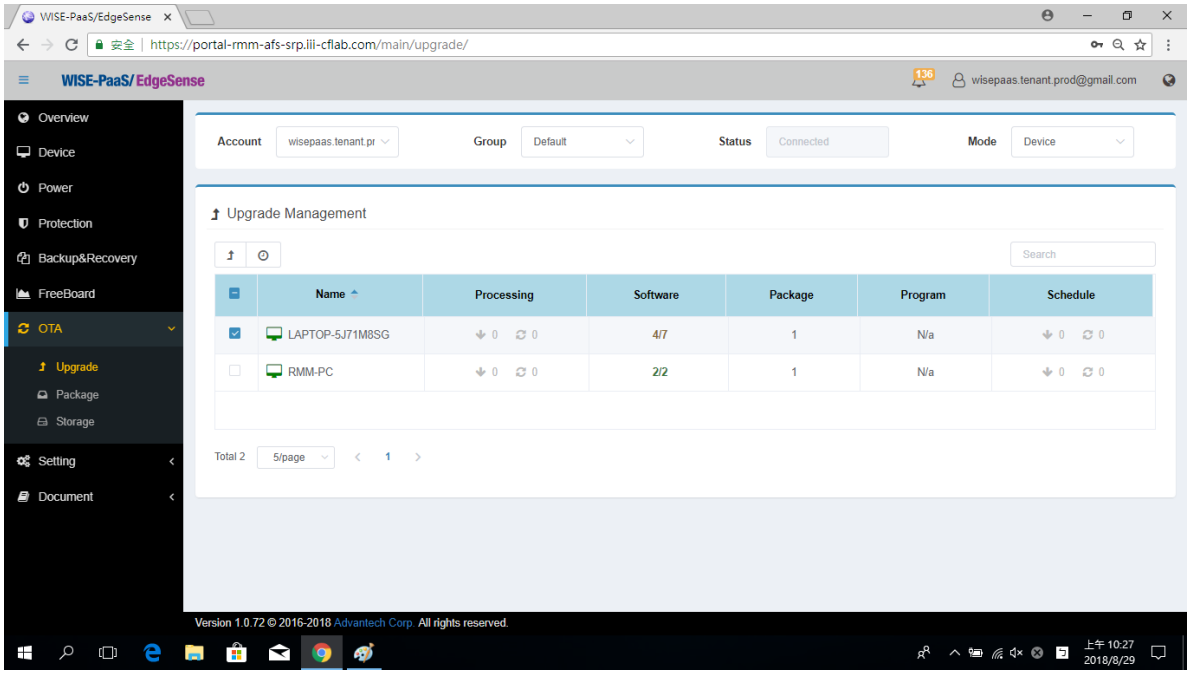


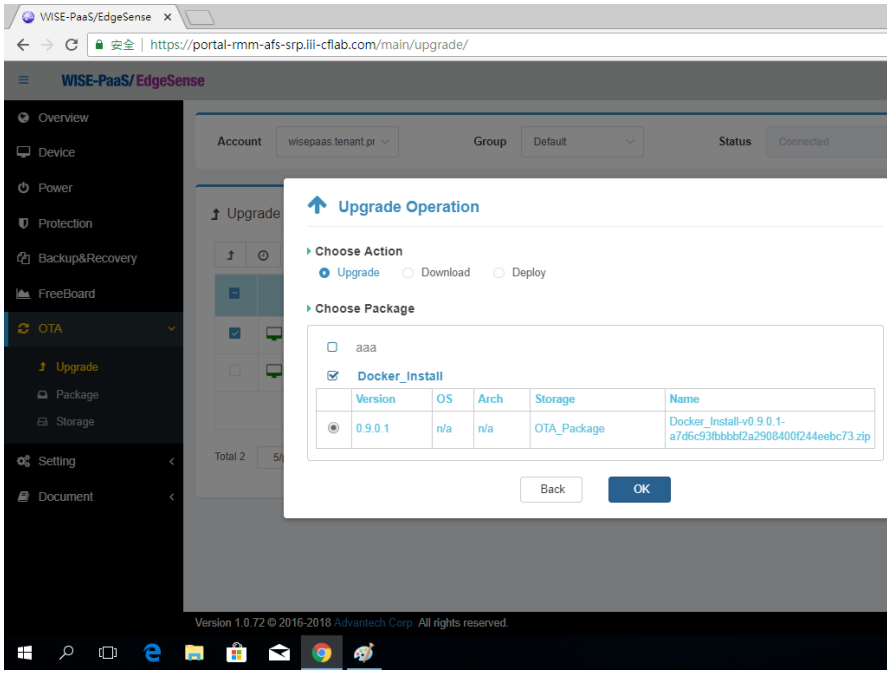
e. Wait a second, when the progress bar goes to 100%, the uploaded file is shown in the list.



3. Send the uploaded file to the edge device for installing automatically.

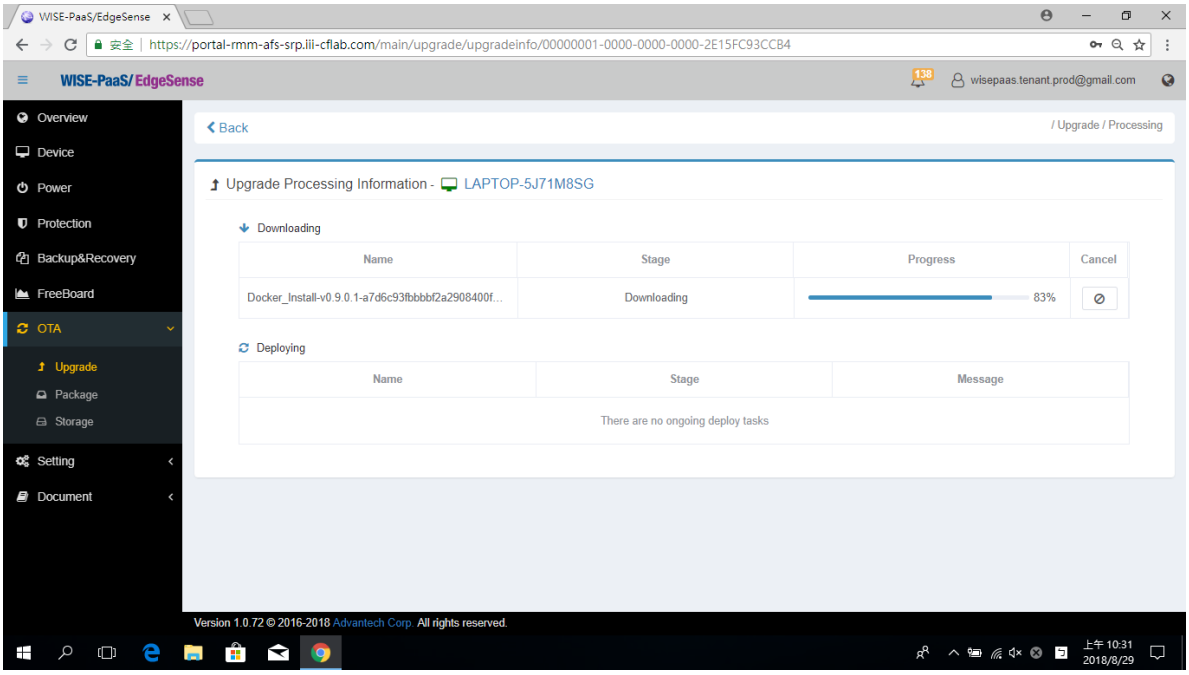
   a. Click “OTA” and “Upgrade”. Then, select the device to be installed.





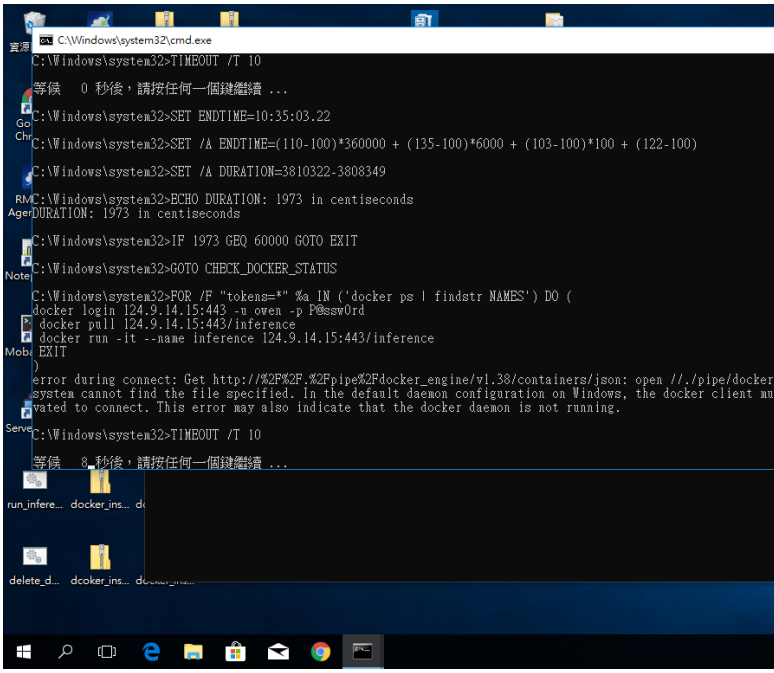
b. Selcet the package which want to **Upgrade**.

c. When the progressing bar goes to 100%, the edge device downloaded the package file completely, and start to



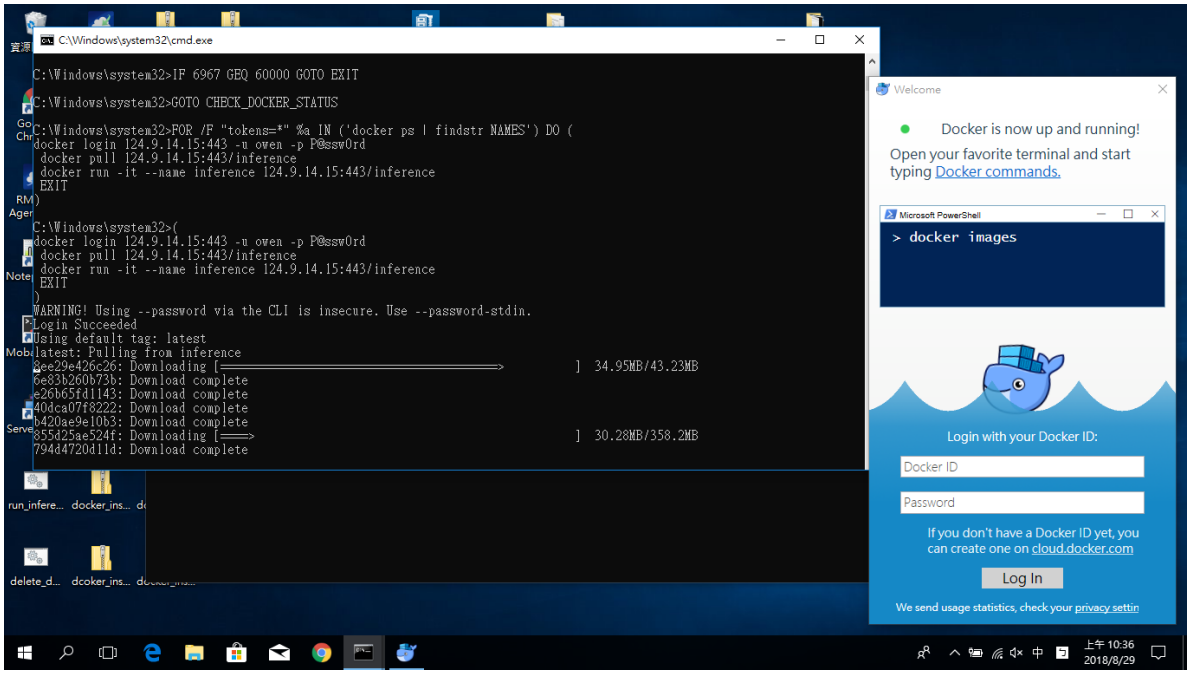
install it.

4. Before installing the package, the edge device restart once. The **Docker** in the edge device starts automatically, and the inference engine runs.



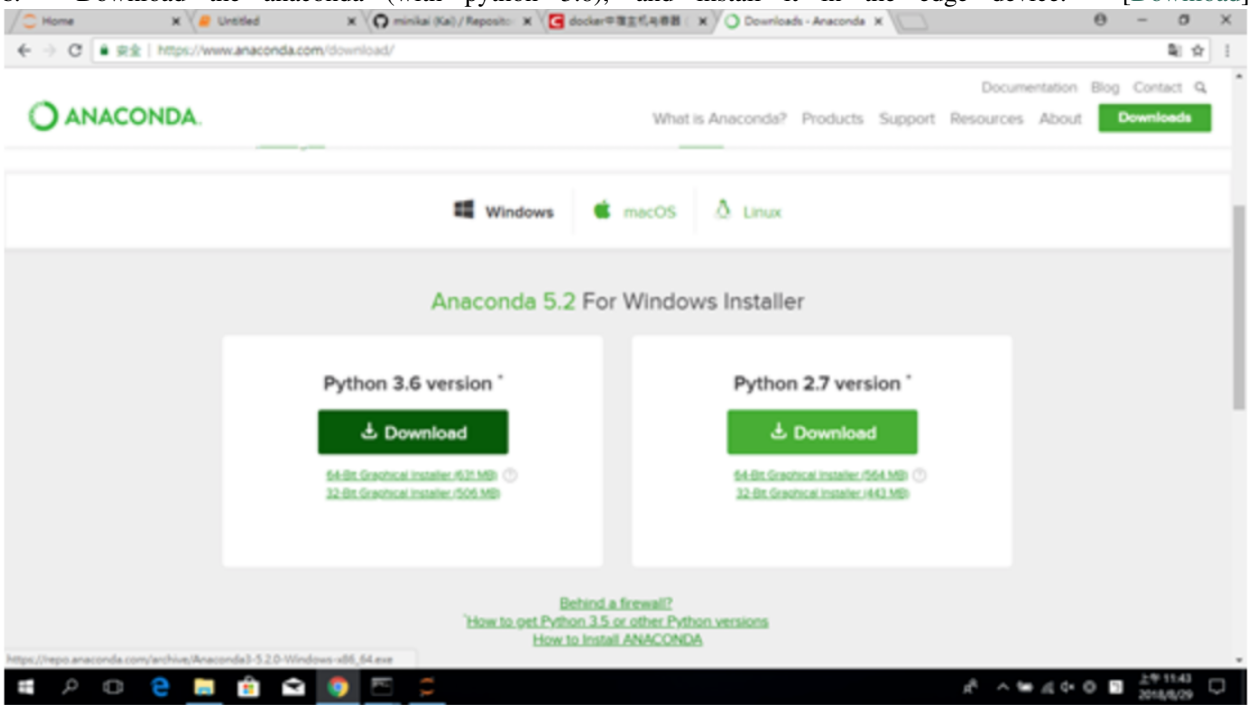
a. The screenshot shows when the installation is running.

b. In the screenshot, it shows the required images are downloading.

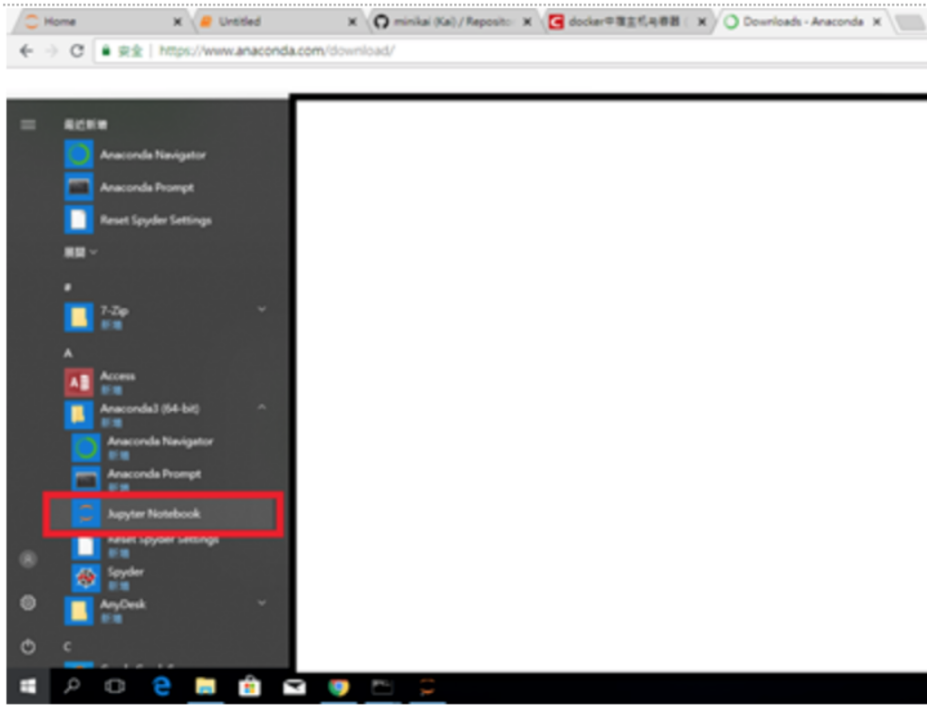


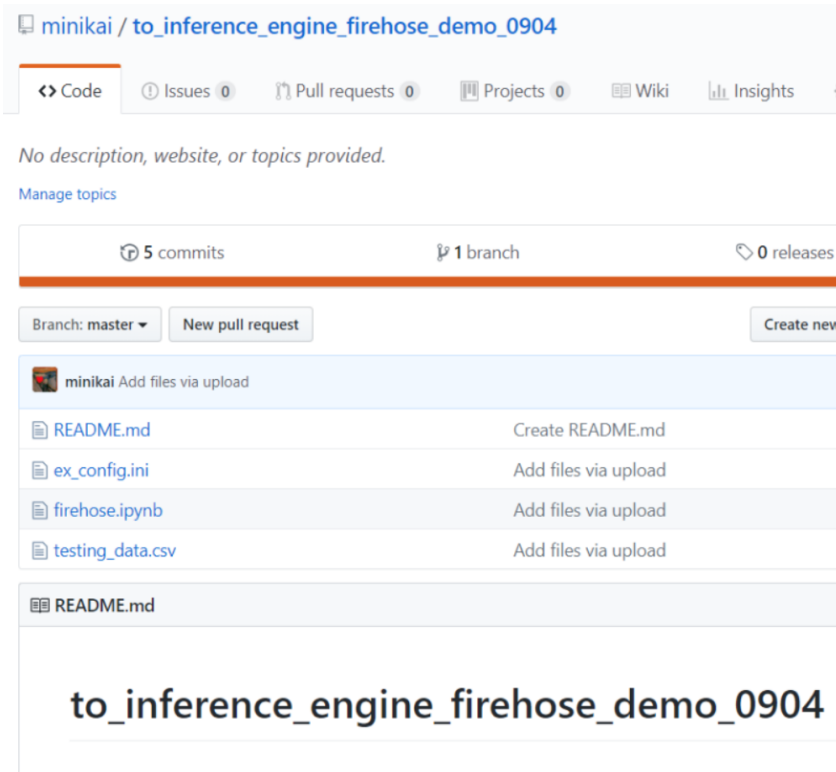
Finally, an edge device has been installed the inference engine automatically. Therefore, if there are many edge devices need to install the inference engine, we just need pick mutiple devices in **Step 3.**, and they will be installed completely.

Now, we can use the model which is trained in Scenario 2. to inference. a. Confirm that the model is trained successfully in Scenario 2., and devivered to edge device by OTA. b. Download the anaconda (with python 3.6), and install it in the edge device. [Download]



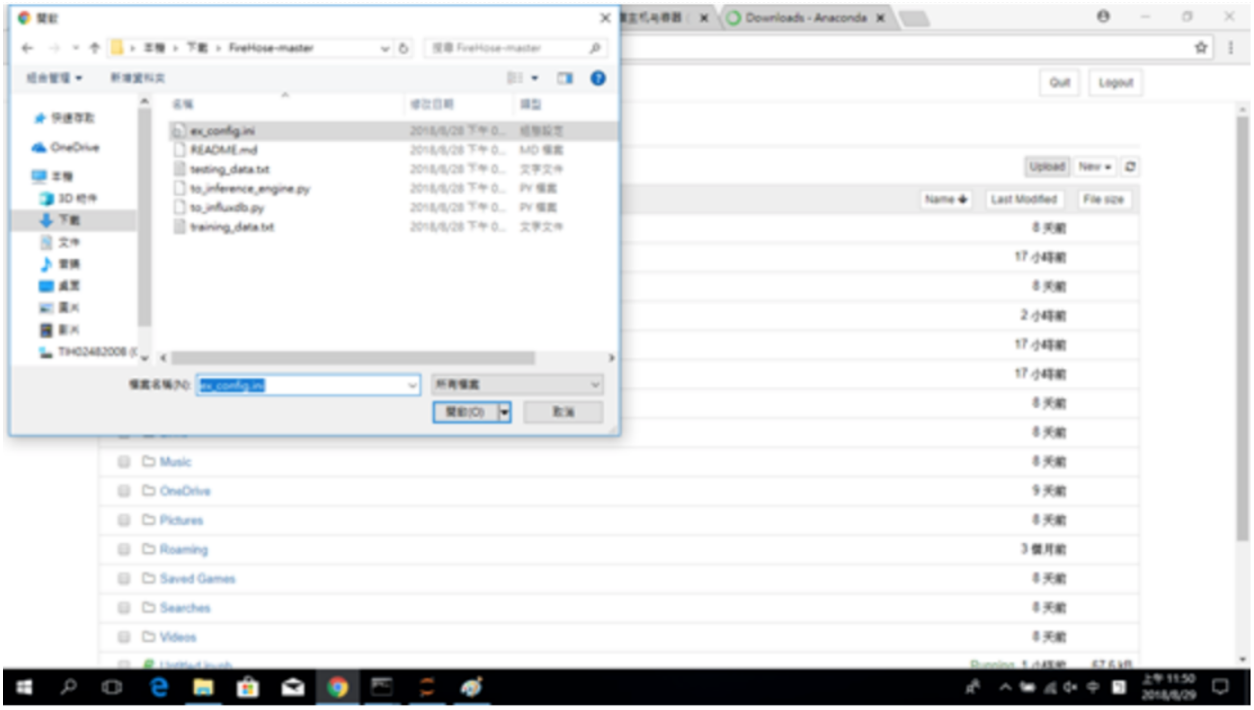
c. Start the **Jupyter Notebook** from application.





d. Download the firehose for testing the inference engine.

e. Click the Upload button at the top right to upload ex_config.ini, firehose.ipynb, and testing_data.csv to jupyter.



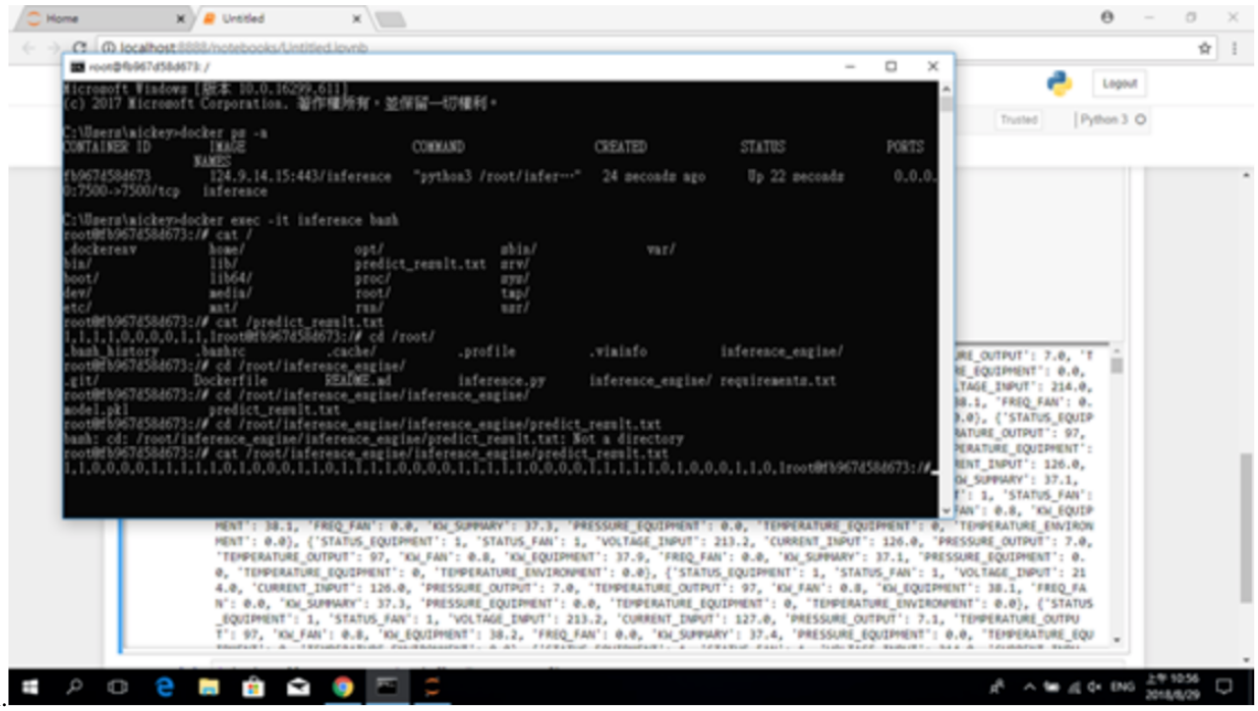
f. Click and modify ex_config.ini, and add “http://127.0.0.1:7500/predict” after “url=”. Then, save the file.

g. Open the firehose.ipynb just uploaded on jupyter and click `Run` to execute.

In [4]:
```
import requests
import json
import pandas as pd
import configparser
import csv
import datetime
import numpy as np
import time



config = configparser.ConfigParser()
config.read('ex_config.ini')
duration_API = config['push_data']['duration_API']
url = config['api']['url']
df1 = pd.read_csv('testing_data.csv')
df2 = df1[['STATUS_FAN','VOLTAGE_INPUT','PRESSURE_OUTPUT','KW_FAN','KW_EQUIPMENT','KW_SUMMARY']]
y = int(df2.shape[0])
i = 1

while i < y :
    row = df1[i:i+1]
    arr = row.values.tolist()
    data={}
    data['STATUS_FAN']=arr[0][1]
    data['VOLTAGE_INPUT']=arr[0][2]
    data['PRESSURE_OUTPUT']=arr[0][3]
    data['KW_FAN']=arr[0][4]
    data['KW_EQUIPMENT']=arr[0][5]
    data['KW_SUMMARY']=arr[0][6]
    data_list=[]
    data_list.append(data)
    json_data = {}
    json_data['data']=data_list
    r = requests.post(url, json=json_data)
    time.sleep(int(duration_API))
    print(json_data)
    i = i+1
```

```
{'data': [{'STATUS_FAN': 0.0, 'VOLTAGE_INPUT': 213.6, 'PRESSURE_OUTPUT': 6.25, 'KW_FAN': 0.15, 'KW_EQUIPME
NT': 25.48, 'KW_SUMMARY': 4.06}]}
{'data': [{'STATUS_FAN': 1.0, 'VOLTAGE_INPUT': 213.6, 'PRESSURE_OUTPUT': 1.85, 'KW_FAN': 0.4, 'KW_EQUIPMEN
```

h. Login to inference_engine, and see the prediction results.i. Execute $ cmd to open the command windowii. Execute $ docker exec -it inference bashiii. In order to check the model is delivered into the inference engine, we can execute $ ls /root/inference_engine/inference_engine/ to see the model.pkl exists or not. (About the model name, it's must named by "model.pkl".)iv. Execute $ cat /root/inference_engine/inference_engine/predict_result.txt to check if the predicted value continues to increase, if the representative is

successful. 
Home
Untitled
localhost:8888/notebooks/Untitled.ipynb
Logout
Trusted
Python 3
root@fb9674584673: /
Microsoft Windows [版本 10.0.16299.611]
(c) 2017 Microsoft Corporation. 著作權所有，並保留一切權利。
C:\Users\aickey>docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
fb9674584673 124.9.14.15:443/inference "python3 /root/infer--" 24 seconds ago Up 22 seconds 0.0.0.0:7500->7500/tcp inference
C:\Users\aickey>docker exec -it inference bash
root@fb9674584673:/# cat /
.dockerenv home/ opt/ sbin/ var/
bin/ lib/ predict_result.txt srv/
boot/ lib64/ proc/ sys/
dev/ media/ root/ tmp/
etc/ mnt/ run/ usr/
root@fb9674584673:/# cat /predict_result.txt
1,1,1,1,0,0,0,0,1,1,1root@fb9674584673:/# cd /root/
.bash_history .bashrc .cache/ .profile .viminfo inference_engine/
root@fb9674584673:/# cd /root/inference_engine/
.git/ Dockerfile README.md inference.py inference_engine/ requirements.txt
root@fb9674584673:/# cd /root/inference_engine/inference_engine/
model.pkl predict_result.txt
root@fb9674584673:/# cd /root/inference_engine/inference_engine/predict_result.txt
bash: cd: /root/inference_engine/inference_engine/predict_result.txt: Not a directory
root@fb9674584673:/# cat /root/inference_engine/inference_engine/predict_result.txt
1,1,0,0,0,0,1,1,1,1,1,1,0,1,0,0,0,1,1,0,1,1,1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,0,1,0,0,0,1,1,0,1root@fb9674584673:/#
ENG

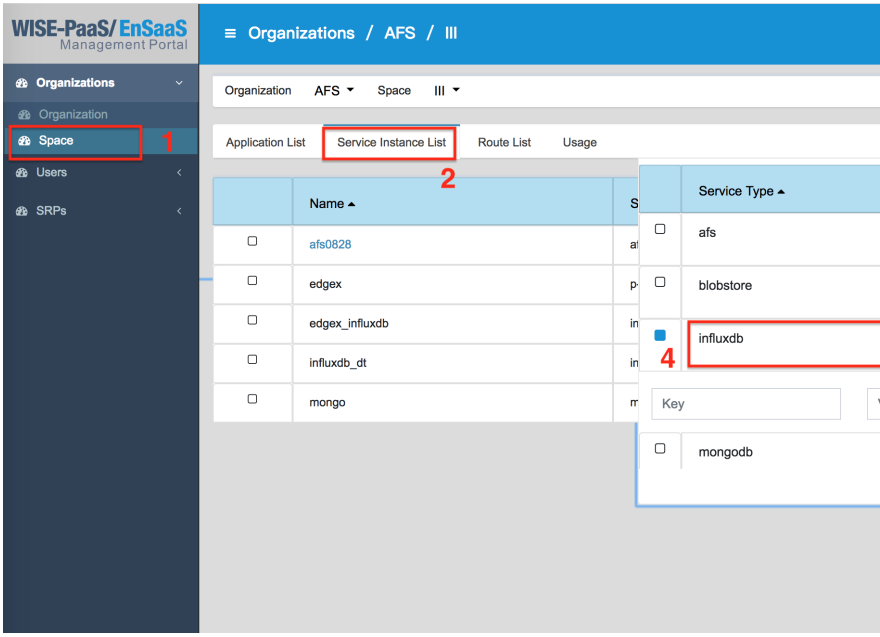
CHAPTER 14

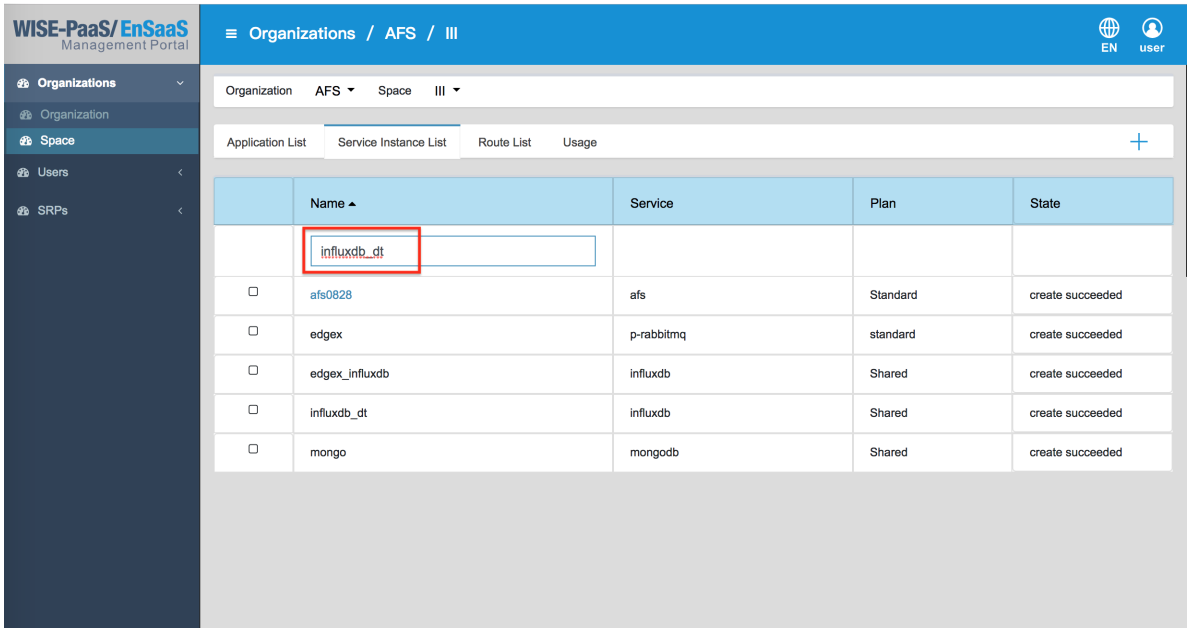
SCENARIO 1. AFS Workspaces - Analytics

14.1 Pre-condition of Analytics

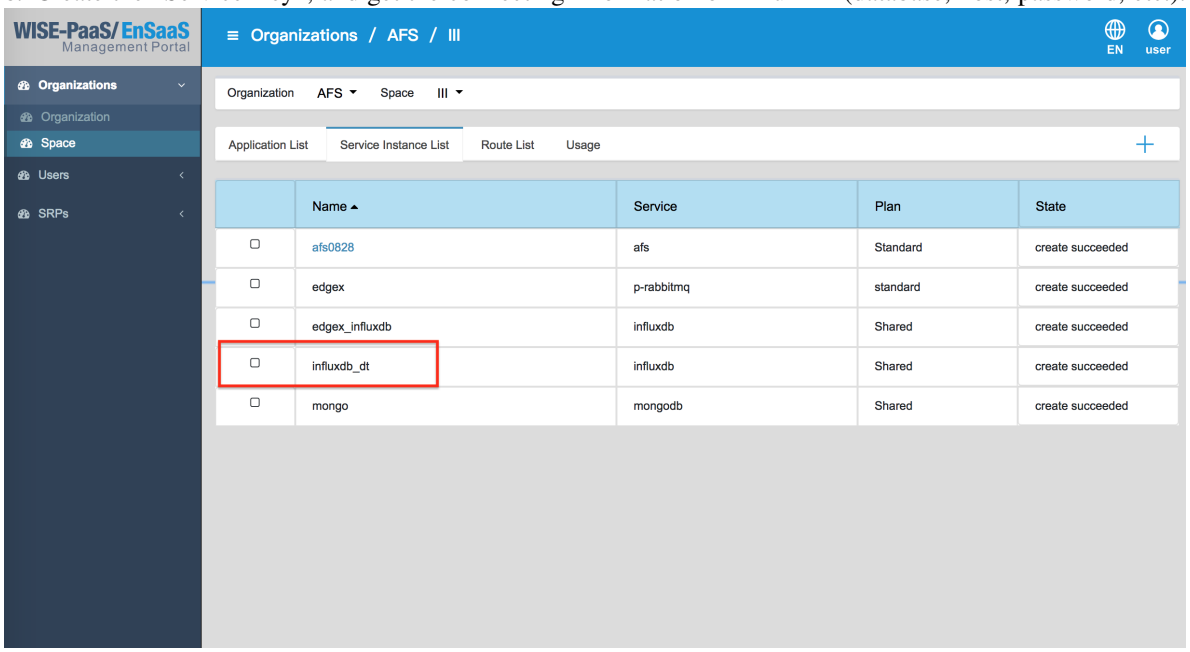
1. Use SSO Tenant/Developer to login Management Portal, and subscribe the InfluxDB service instance. (Please refer Management Portal User Manual.)

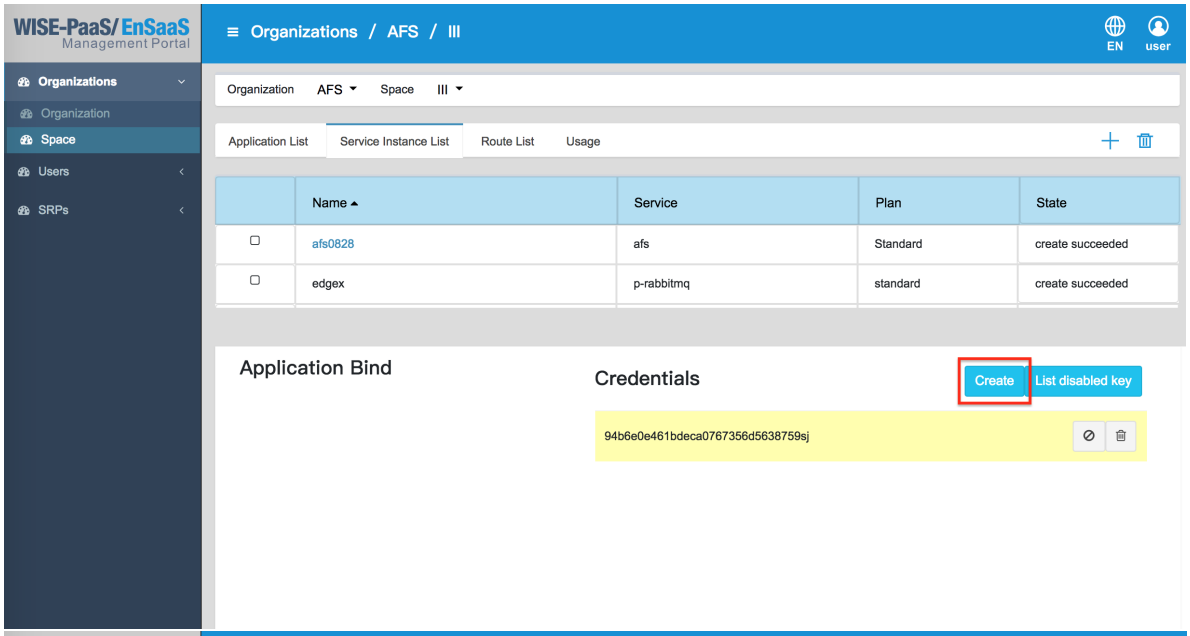
   a. Subscribe the service and name influxdb_dt.

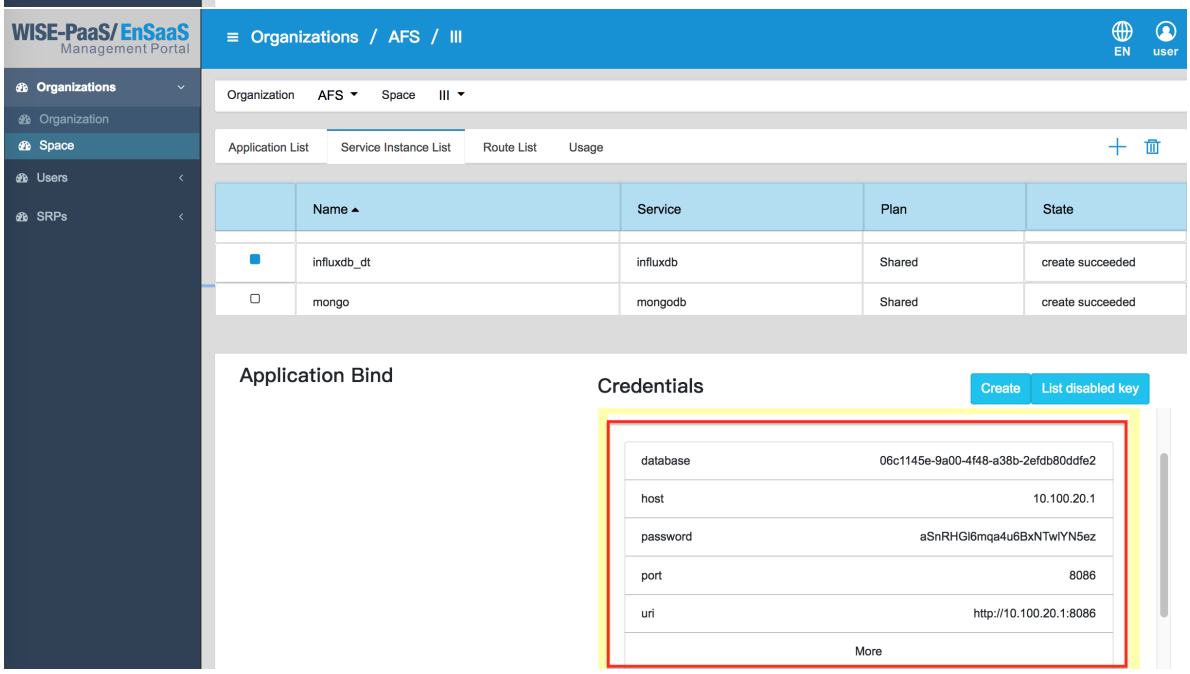




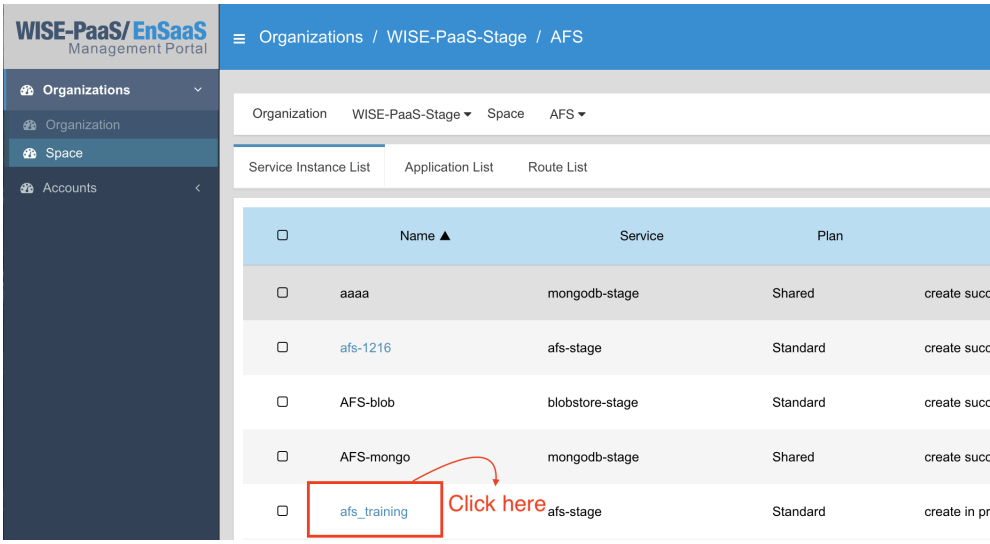
b. Create the “Service Key”, and get the connecting information of InfluxDB (database, host, password, etc.).





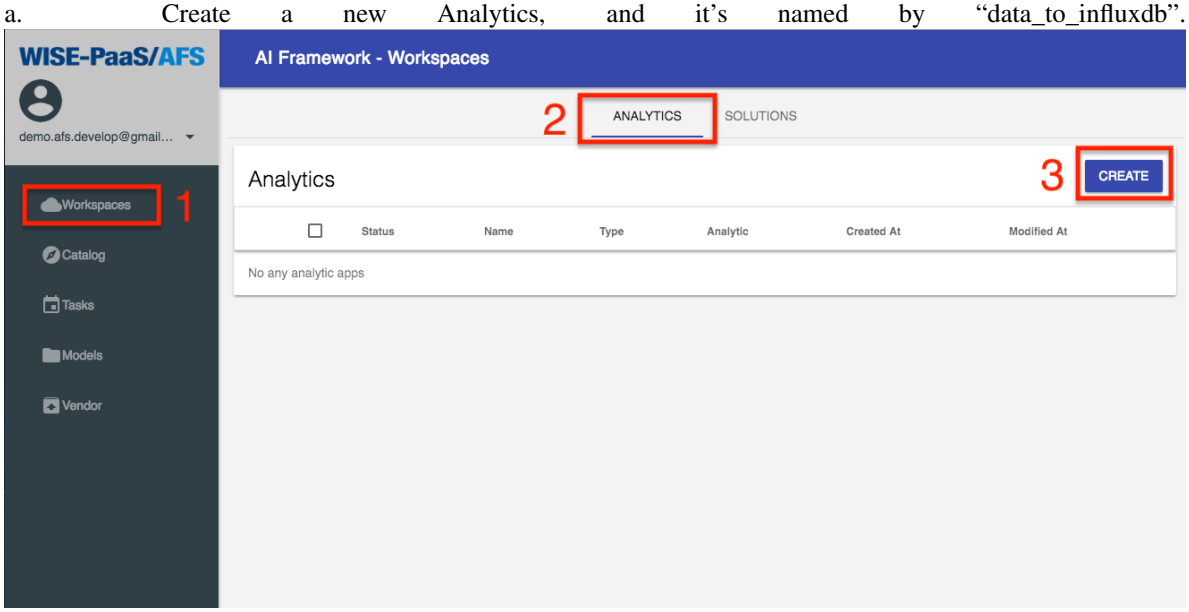


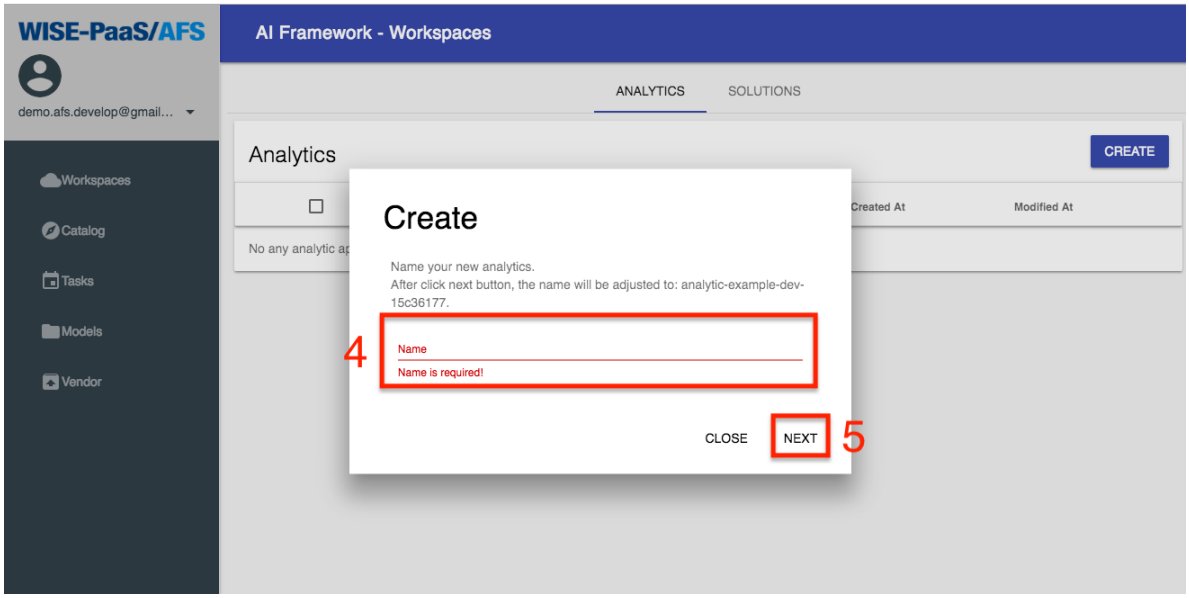
2. Subscribe the AFS service instance from Management Portal, and it's named by afs_training. When it shows **create succeeded**, the AFS service instance is created.



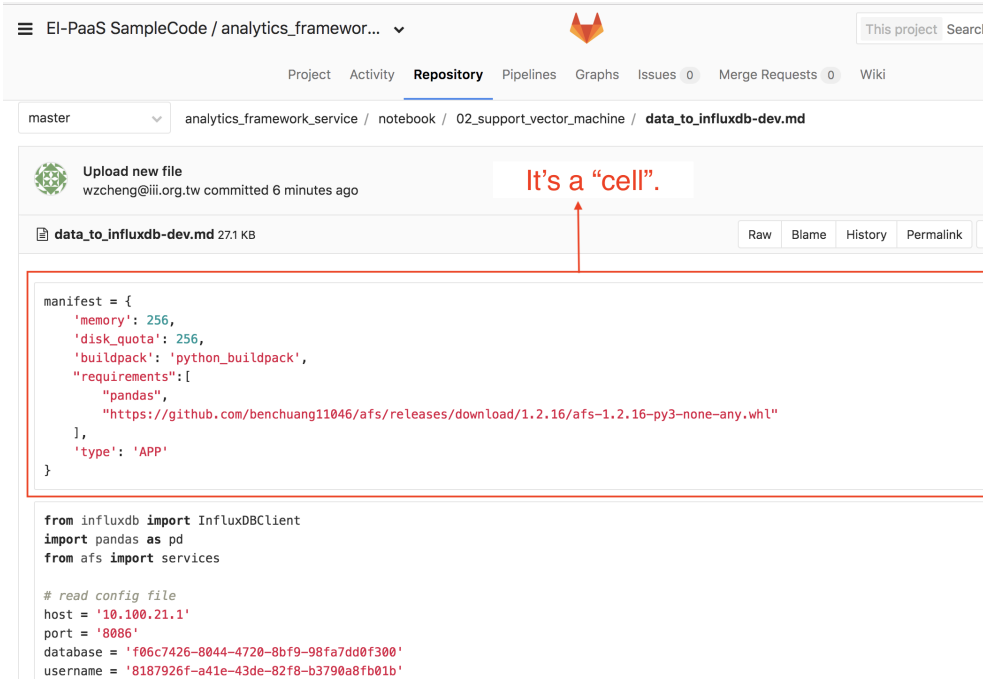
3. Click **afs_training** to enter the AFS.
4. Create a new Analytics, Firehose, to upload the training data to database.

   a. Create a new Analytics, and it's named by "data_to_influxdb".

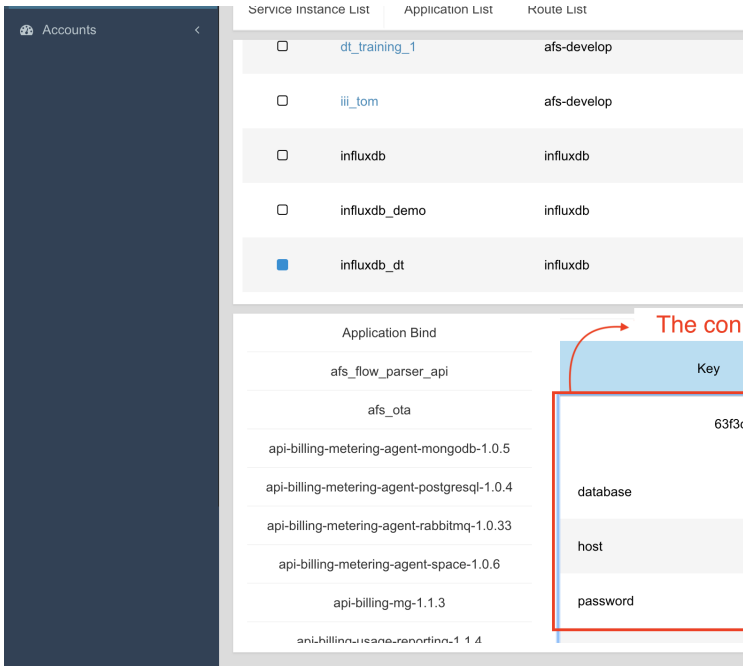




b. Copy the sample code to the data_to_influxdb, and the code must be divided by cell.



Note: The **Cell** is defined as follows:



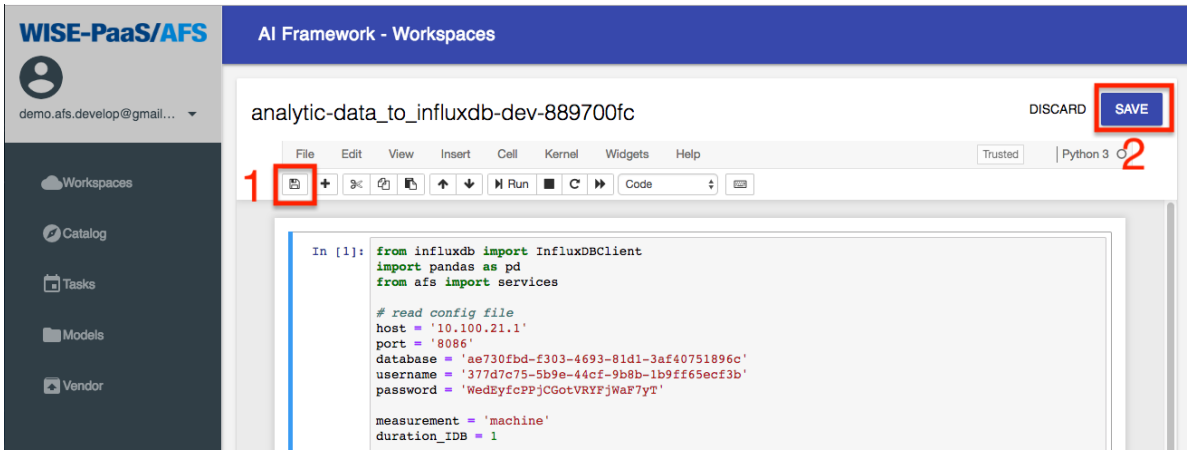
c. Enter the connecting information to the data_to_influxdb.

```
# read config file
# afs_ser = services()
# credential = afs_ser.get_service_info()['influxdb'][0]

host = ''
port = ''
database = ''
username = ''
password = ''
measurement = 'data_mea'
duration_IDB = 5
client = InfluxDBClient(host, port, username, password, database)  # connect influxdb
while True:
    df = pd.read_csv('training_data.csv')
    data = {}
    for i in range(0, len(df)):
        data['measurement'] = measurement
        tags = {}
        tags['sn'] = 'system_data'
        data['tags'] = tags
        fields = {}
        fields['STATUS_FAN']= int(df.iloc[i]['STATUS_FAN'])
        fields['STATUS_EQUIPMENT']= int(df.iloc[i]['STATUS_EQUIPMENT'])
        fields['TEMPERATURE_OUTPUT']= int(df.iloc[i]['TEMPERATURE_OUTPUT'])
        fields['PRESSURE_OUTPUT']= df.iloc[i]['PRESSURE_OUTPUT']
        fields['VOLTAGE_INPUT']= df.iloc[i]['VOLTAGE_INPUT']
        fields['KW_EQUIPMENT']= df.iloc[i]['KW_EQUIPMENT']
        fields['KW_FAN']= df.iloc[i]['KW_FAN']
        fields['KW_SUMMARY']= df.iloc[i]['KW_SUMMARY']
        fields['TEMPERATURE_ENVIRONMENT']= df.iloc[i]['TEMPERATURE_ENVIRONMENT']
        data['fields'] = fields
```

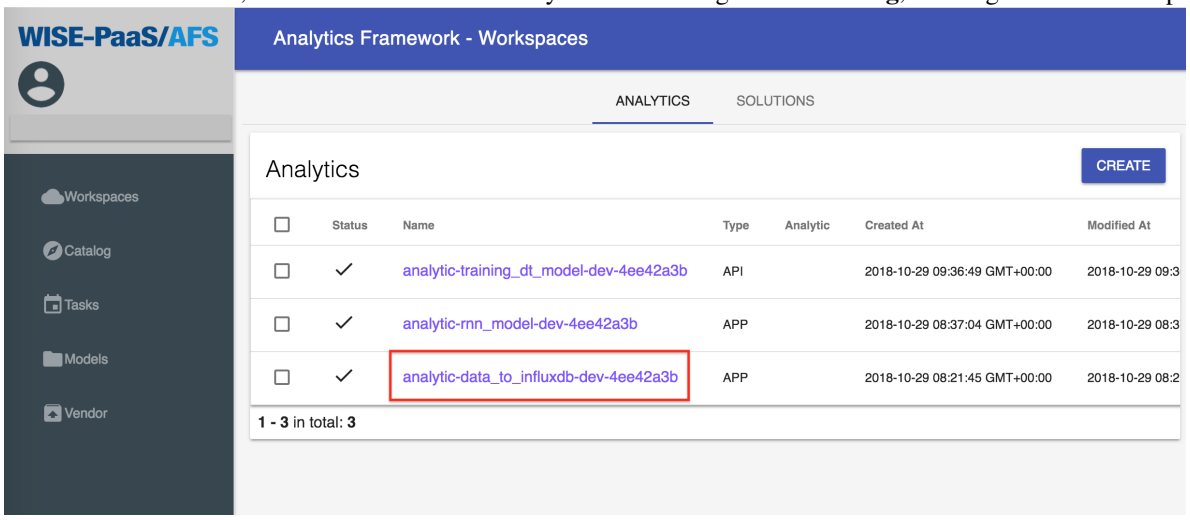
Enter the connecting information of influxdb.

d. Execute each cell.

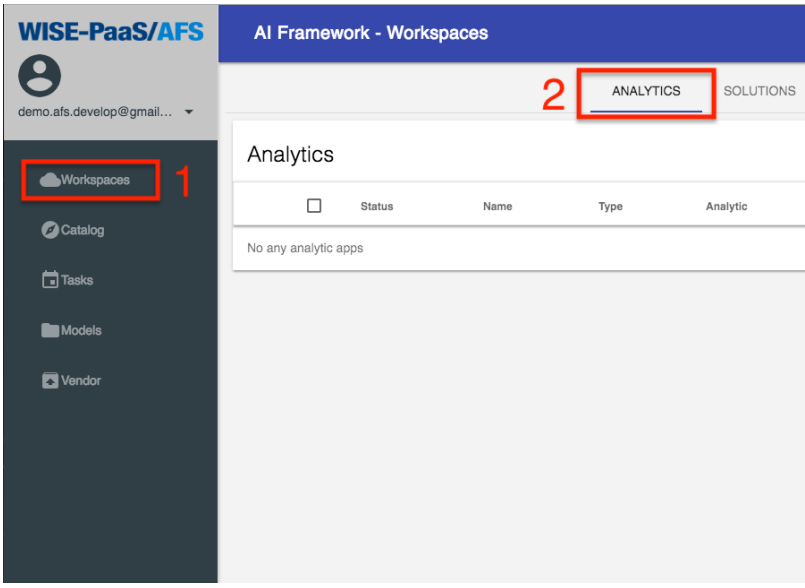
e. Click the icon in the left side to save it, and click `SAVE` to upload the Analytics App.



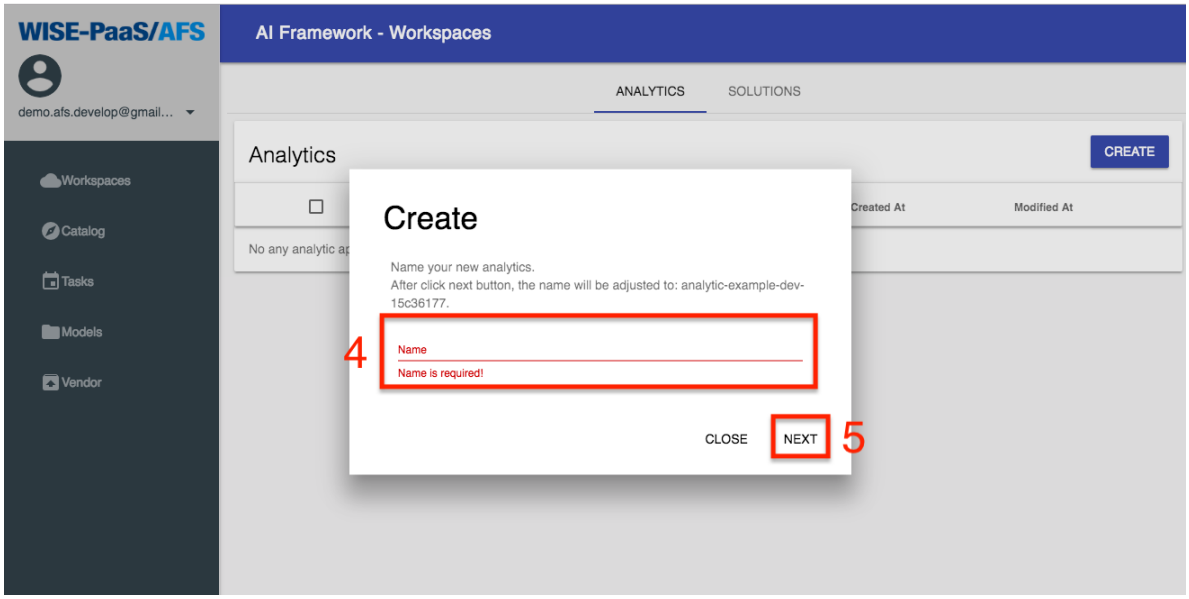
f. Wait a minute, the status of the Analytic will change to **Running**, and go to next step.



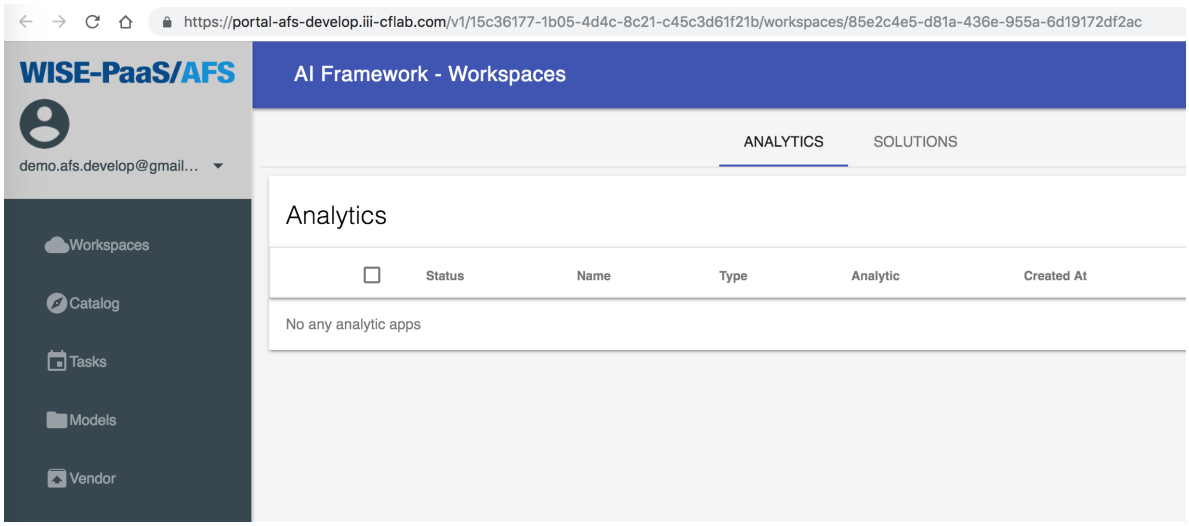
14.2 Create Analytics by Online Code IDE



1. Create a new Analytics, and it's named by rnn_model.

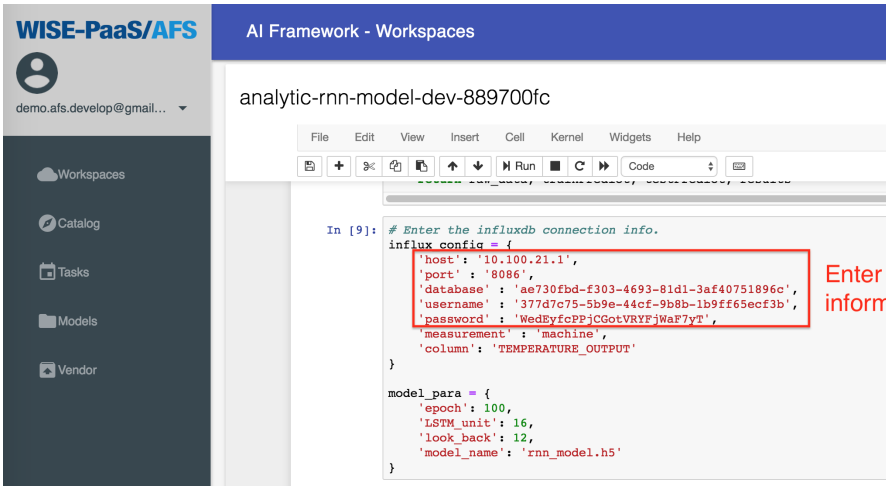


2. Copy the sample code to rnn-model which is created in last step.
3. Install the scikit-learn package, please copy the command from the link, and paste the code in a new cell as the follows. After executing the cell, delete it.


https://portal-afs-develop.iii-cflab.com/v1/15c36177-1b05-4d4c-8c21-c45c3d61f21b/workspaces/85e2c4e5-d81a-436e-955a-6d19172df2ac
WISE-PaaS/AFS
AI Framework - Workspaces
demo.afs.develop@gmail...
ANALYTICS
SOLUTIONS
Workspaces
Catalog
Tasks
Models
Vendor
Analytics
Status
Name
Type
Analytic
Created At
No any analytic apps

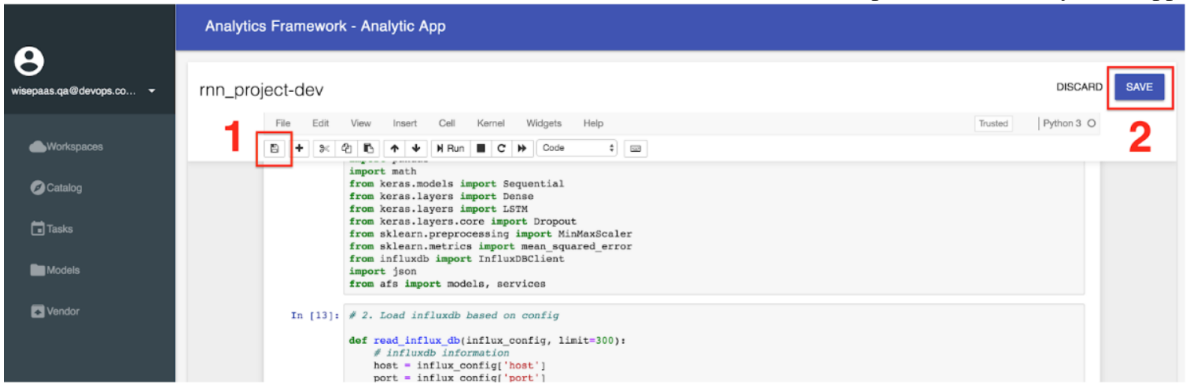

https://portal-afs-develop.iii-cflab.com/v1/15c36177-1b05-4d4c-8c21-c45c3d61f21b/workspaces/85e2c4e5-d81a-436e-955a-6d19172df2ac
Use the example url to substitute.
!pip install https://portal-afs-develop.iii-arfa.com/v1/2e94bf7f-307e-4c55-a3f7-e30c81319160/workspaces/6dcc2e50-afce-408d-9c38-ec5dd1a48098/vendor/urllib3-1.2
!pip install https://portal-afs-develop.iii-arfa.com/v1/2e94bf7f-307e-4c55-a3f7-e30c81319160/workspaces/6dcc2e50-afce-408d-9c38-ec5dd1a48098/vendor/idna-2.7-py
!pip install https://portal-afs-develop.iii-arfa.com/v1/2e94bf7f-307e-4c55-a3f7-e30c81319160/workspaces/6dcc2e50-afce-408d-9c38-ec5dd1a48098/vendor/chardet-3.0
!pip install https://portal-afs-develop.iii-arfa.com/v1/2e94bf7f-307e-4c55-a3f7-e30c81319160/workspaces/6dcc2e50-afce-408d-9c38-ec5dd1a48098/vendor/certifi-201
!pip install https://portal-afs-develop.iii-arfa.com/v1/2e94bf7f-307e-4c55-a3f7-e30c81319160/workspaces/6dcc2e50-afce-408d-9c38-ec5dd1a48098/vendor/pytz-2018.5
!pip install https://portal-afs-develop.iii-arfa.com/v1/2e94bf7f-307e-4c55-a3f7-e30c81319160/workspaces/6dcc2e50-afce-408d-9c38-ec5dd1a48098/vendor/six-1.11.0-
!pip install https://portal-afs-develop.iii-arfa.com/v1/2e94bf7f-307e-4c55-a3f7-e30c81319160/workspaces/6dcc2e50-afce-408d-9c38-ec5dd1a48098/vendor/requests-2.
!pip install https://portal-afs-develop.iii-arfa.com/v1/2e94bf7f-307e-4c55-a3f7-e30c81319160/workspaces/6dcc2e50-afce-408d-9c38-ec5dd1a48098/vendor/python_date
!pip install https://portal-afs-develop.iii-arfa.com/v1/2e94bf7f-307e-4c55-a3f7-e30c81319160/workspaces/6dcc2e50-afce-408d-9c38-ec5dd1a48098/vendor/influxdb-5.
!pip install https://portal-afs-develop.iii-arfa.com/v1/2e94bf7f-307e-4c55-a3f7-e30c81319160/workspaces/6dcc2e50-afce-408d-9c38-ec5dd1a48098/vendor/afs-1.2.16.

4. Enter the connecting information of InfluxDB.

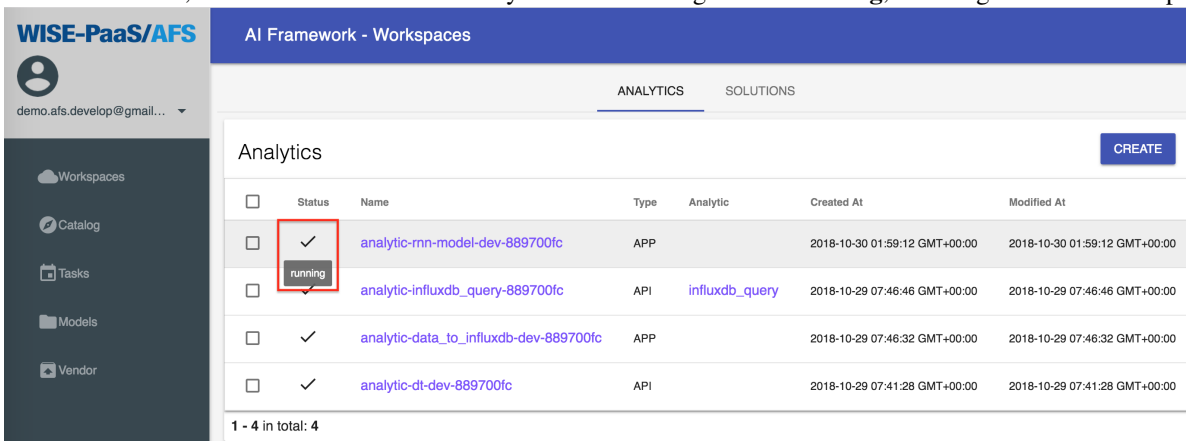


5. Execute all of cells.

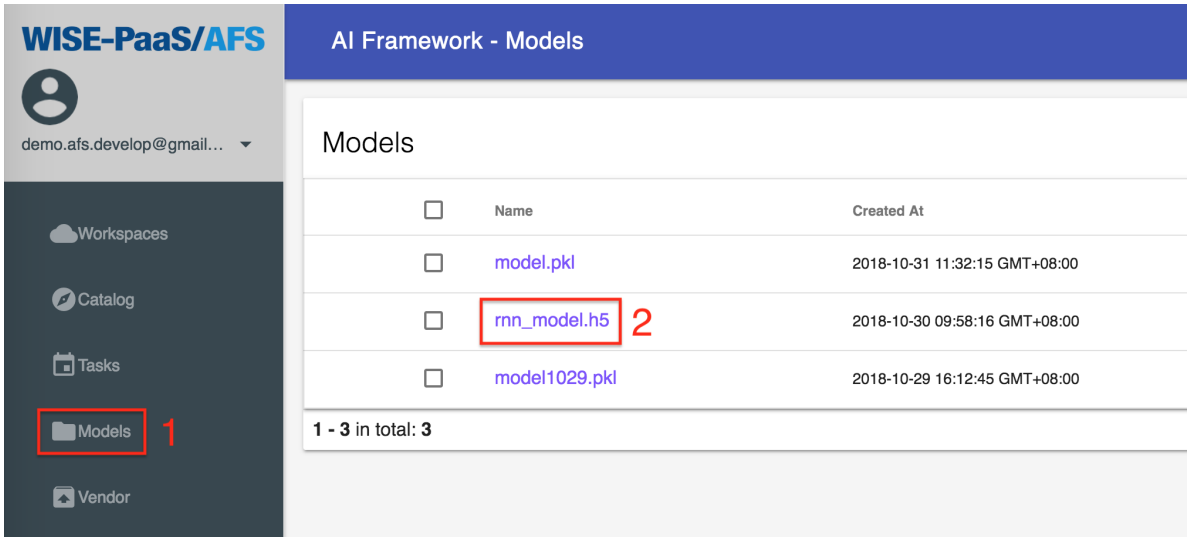
6. Click the icon in the left side to save it, and click `SAVE` to upload the Analytics App.



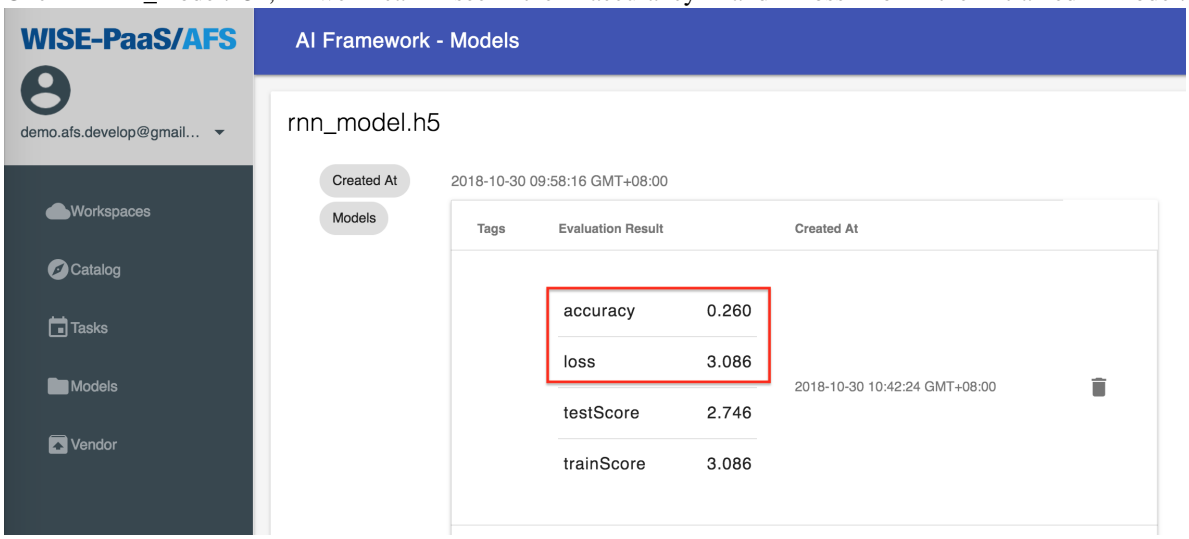
7. Wait a minute, the status of the Analytics will change to **Running**, and go to next step.



8. Click **Models** in the menu, the model repository which is named by “rnn_model.h5” is created.



9. Click "rnn_model.h5", we can see the accurancy and loss of the trained model.



CHAPTER 15

SCENARIO 2. AFS Workspaces - Solutions

Create Online Flow IDE in the **AFS Workspaces - Solutions**, and train the **Desicion Tree** model. After training the model, use the OTA to deliver the model to the edge device.

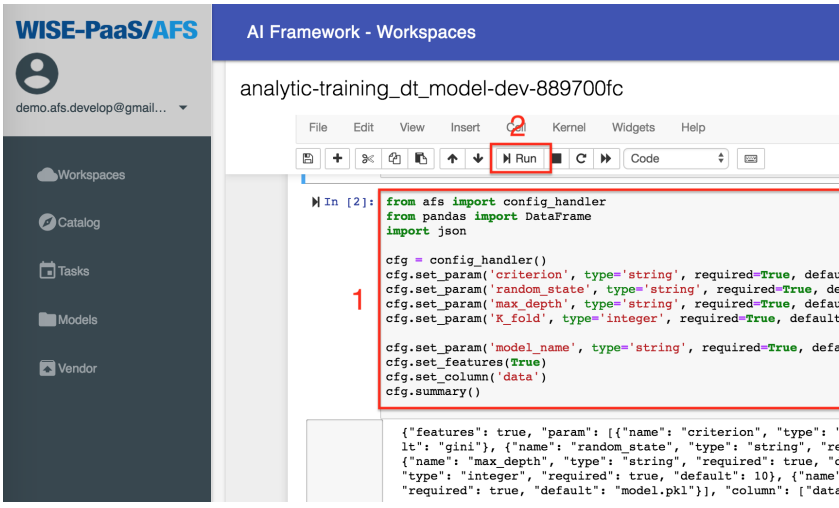
15.1 Pre-condition of Solutions

1. Create the Decision Tree node in the Online Flow IDE.

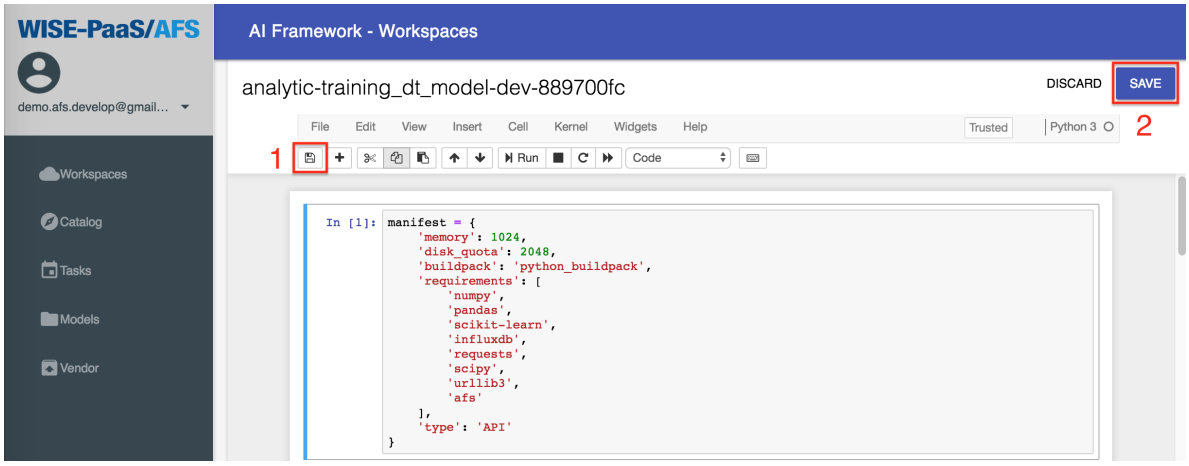
   a. Create a new Aanlytic, and it's named by training_dt_model. About the detail process, please refer the Pre-condition Step 4.b in the Scenario 1.

   b. Copy the sample code to training_dt_model, and the code must be divided by cell.

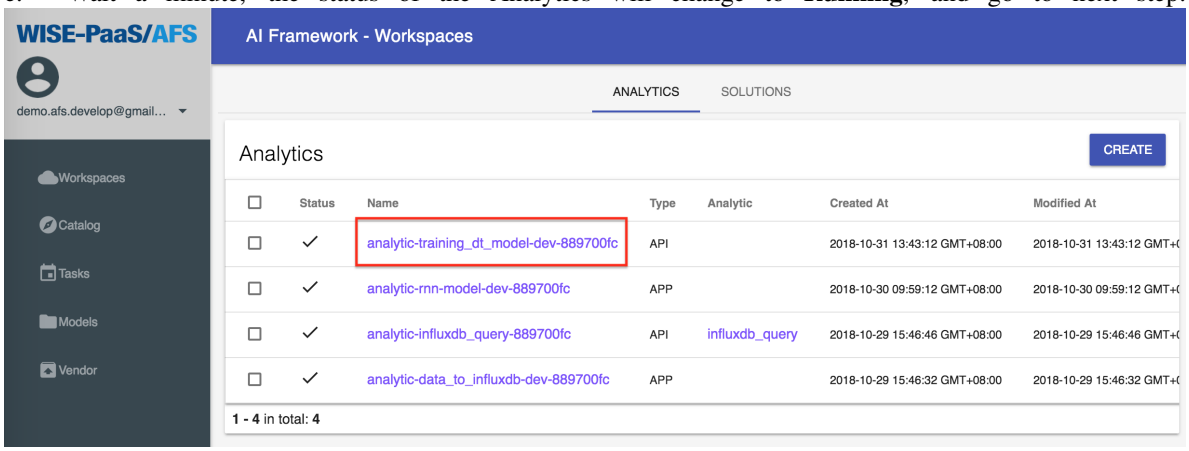
   c. Pick the second cell, and click `Run` to execute it.



   d. Click the icon in the left side to save it, and click `SAVE` to upload the Analytics App.



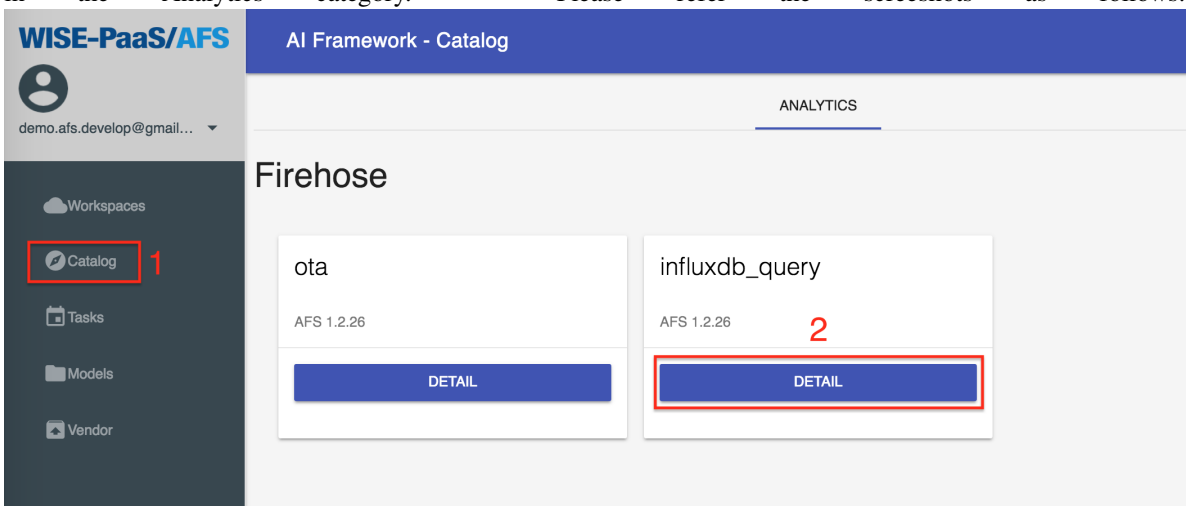
e. Wait a minute, the status of the Analytics will change to **Running**, and go to next step.

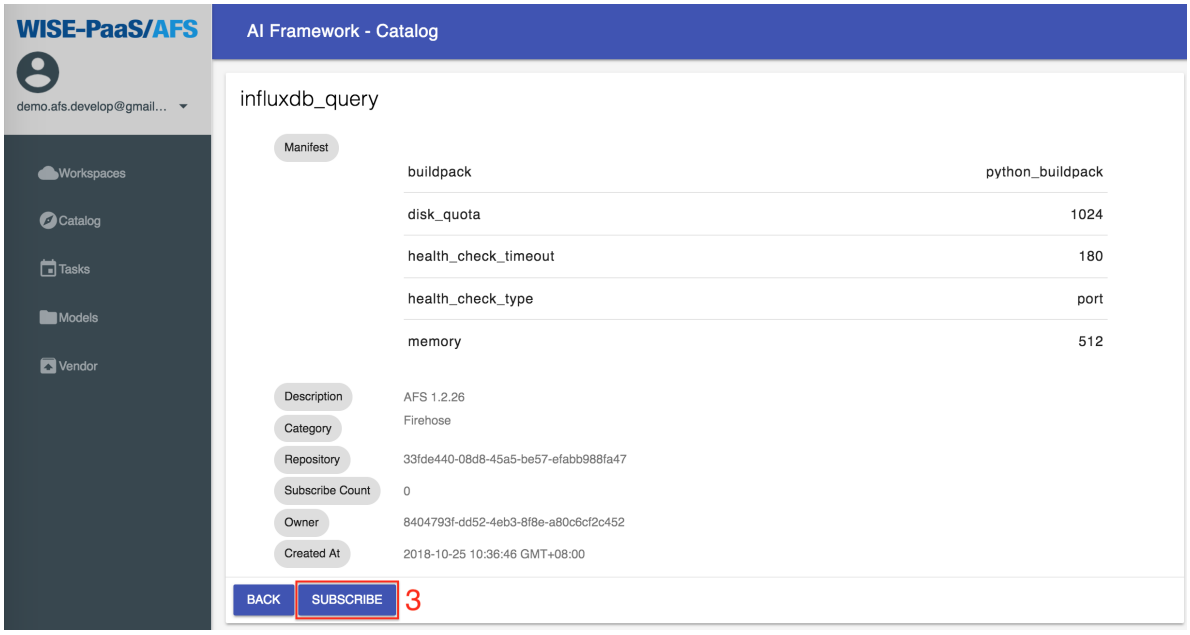


Note: After the processing, the training_dt_model node is generated in the Online Flow IDE.

2. Subscribe the influxdb_query node in the Online Flow IDE.

a. In the **Catalog**, we can subscribe the **influxdb_query** node in the Analytics category. Please refer the screeshots as follows:





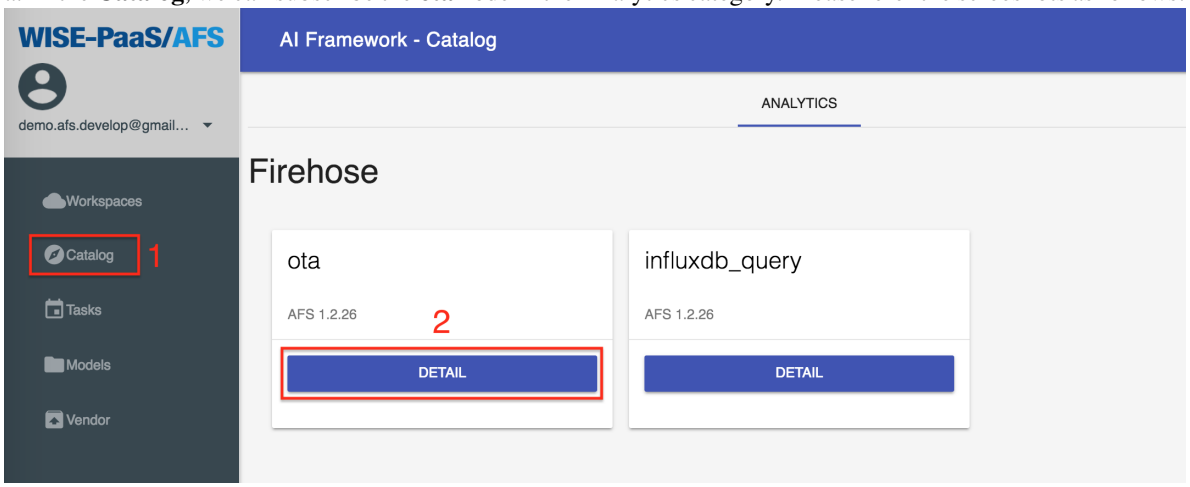
b. The **influxdb_query** is shown in the **Analytics List** when it's subscribed successfully.

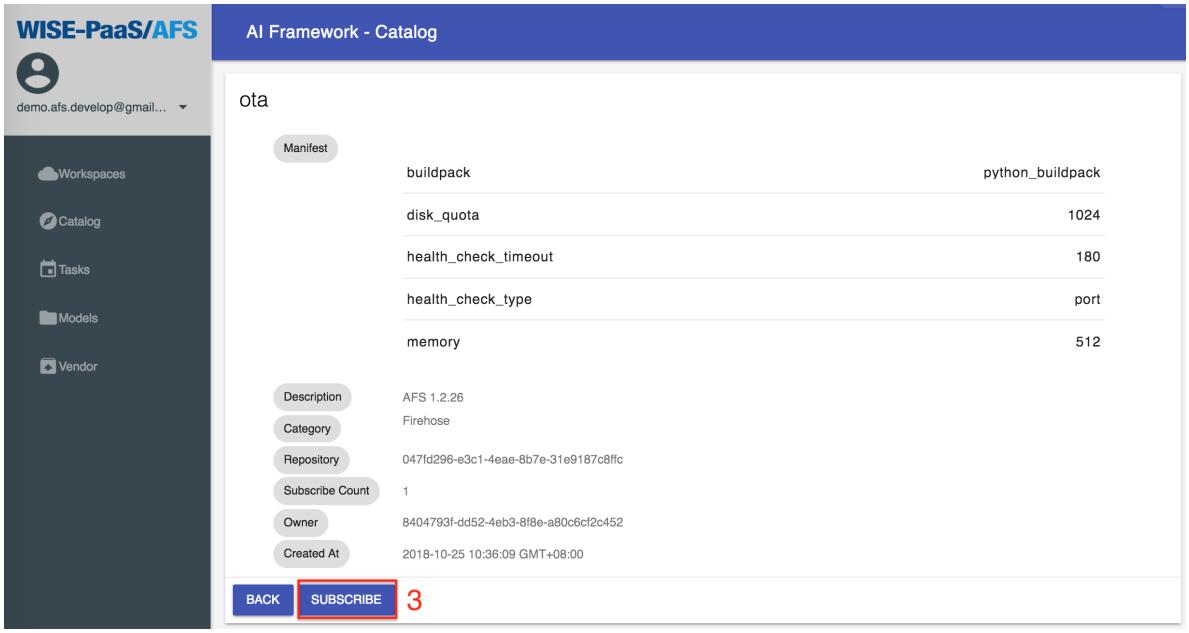
c. Wait a minute, the status of the Analytics will change to **Running**, and go to next step.

Note: After the processing, the influxdb_query node is generated in the Online Flow IDE.

3. Subscribe the ota node in the Online Flow IDE.

a. In the **Catalog**, we can subscribe the **ota** node in the Analytics category. Please refer the screeshots as follows:





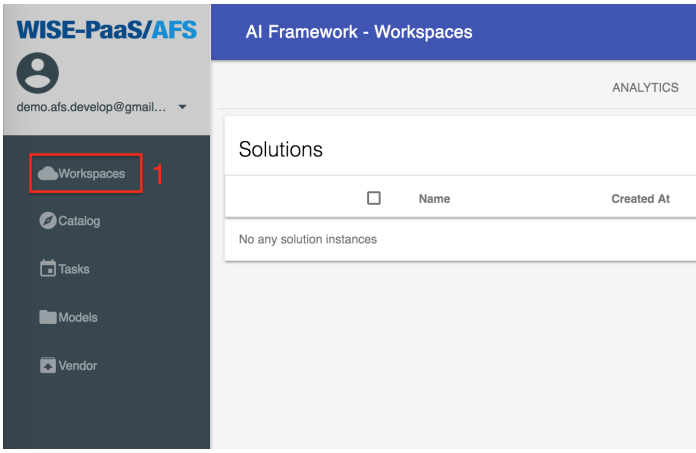
b. The **ota** is listed in the **Analytics** when it's subscribed successfully.

c. Wait a minute, the status of the Analytics will change to **Running**, and go to next step.

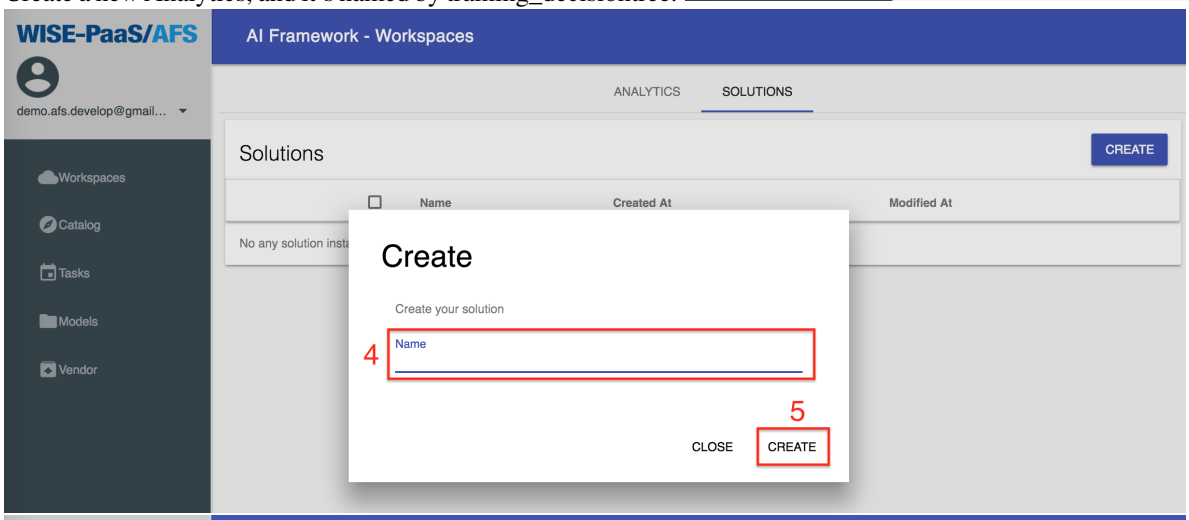
Note: After the processing, the ota node is generated in the Online Flow IDE.

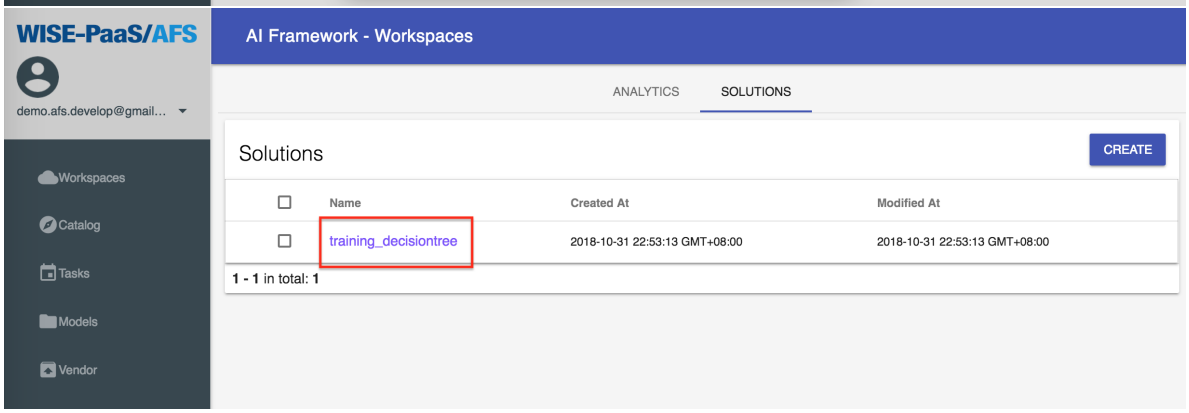
4. Setup the **RMM** device, include (1) install the **RMM Agent** in the edge device; (2) register the device; and (3) create a storage for RMM. Please refer the document.

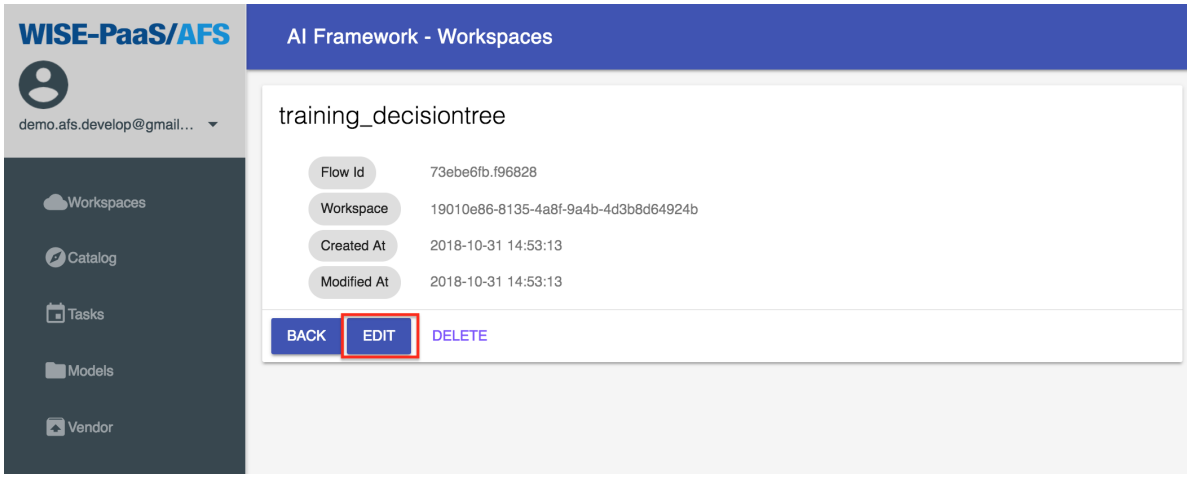
15.2 Create Solution by Online Flow IDE



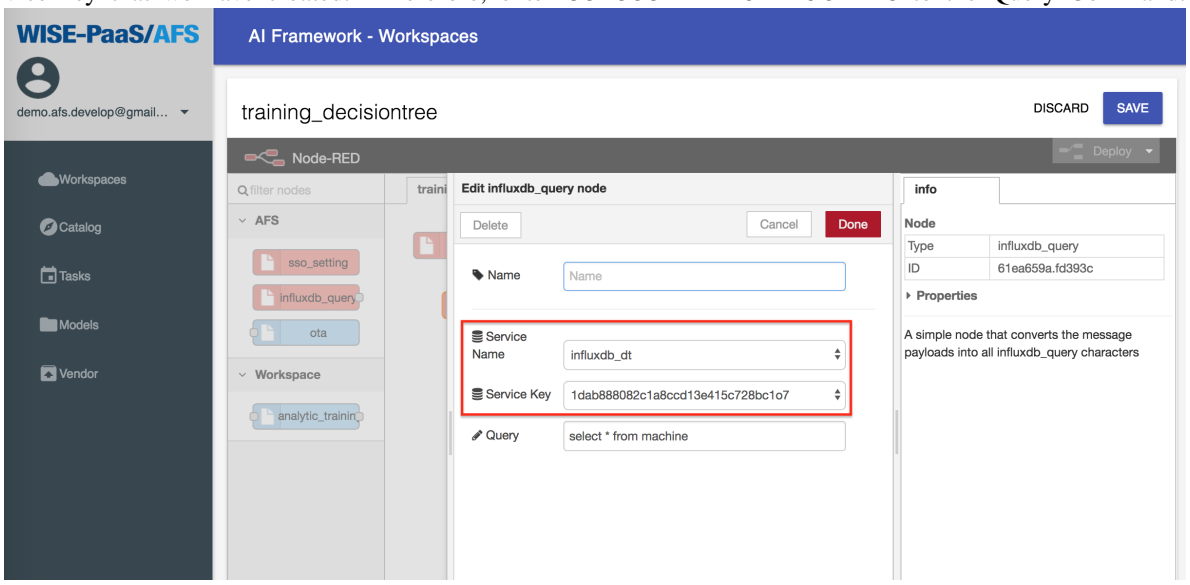
1. Create a new Analytics, and it's named by training_decisiontree.



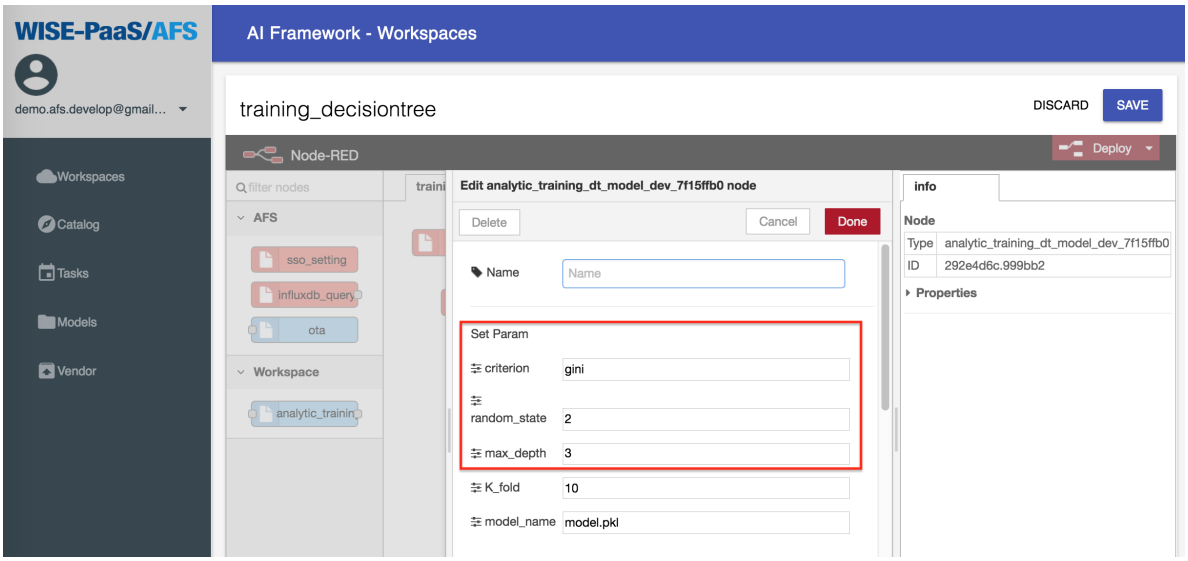




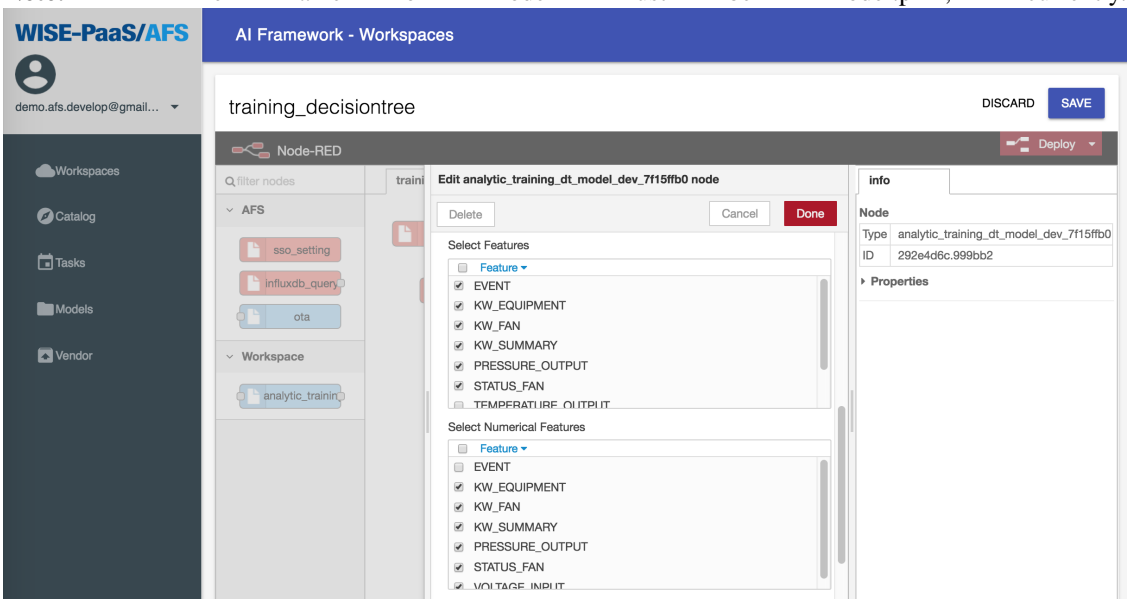
2. Pull the sso_setting node from the list in the left side. Then, enter the SSO Username and SSO Password in it.
3. Pull the influxdb_query node from the list in the left side. Then, select the influxdb_dt and the service key that we have created. Therefore, enter `select * from machine` to the Query Command.



4. Pull the training_dt_model node from the list in the left side, and setup the parameters.



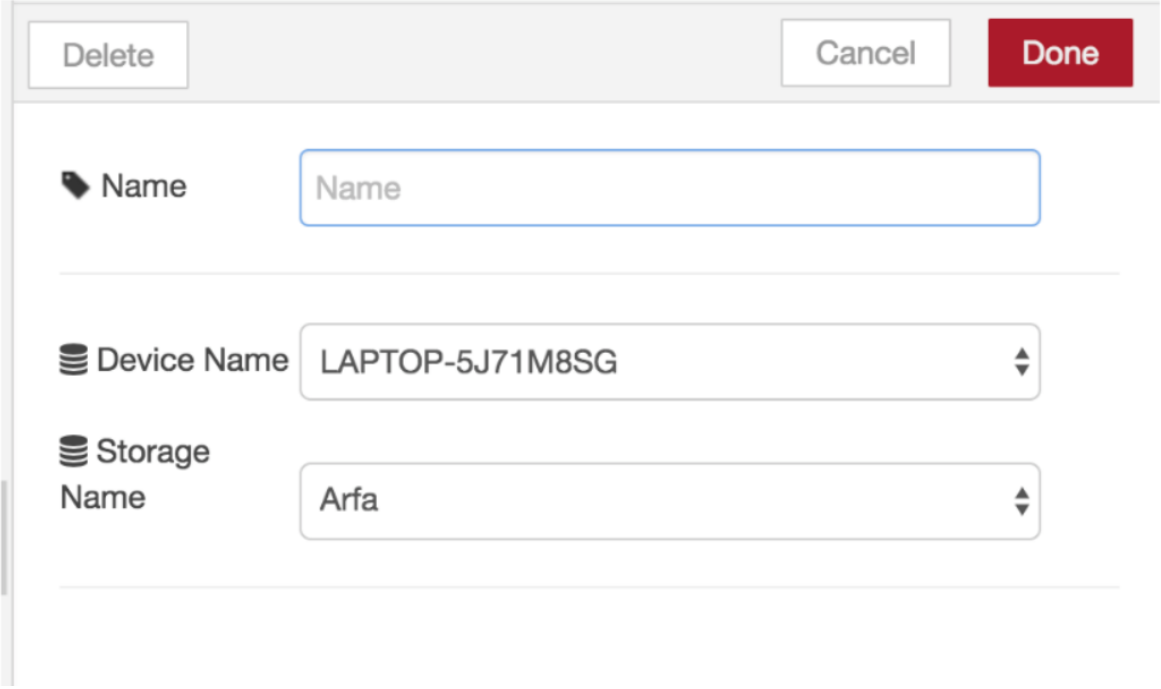
- criterion: Can't be empty. Please enter *gini* or *entropy*, separated by commas, and without spaces between parameters and commas.
- random_state and max_depth: Enter the integer only. If want to optimize the parameters, we can fill in multiple sets of parameters in the random_state and max_depth fields as shown above. The parameters must be separated by commas. There must be no blank between the parameters and the comma.
- K_fold: Enter the times for cross validation, and it must be an interger and bigger than one.
- model_name: Name the trained model, must .pkl type(e.g., model.pkl). **Note:** The name of model must be "model.pkl", currently.



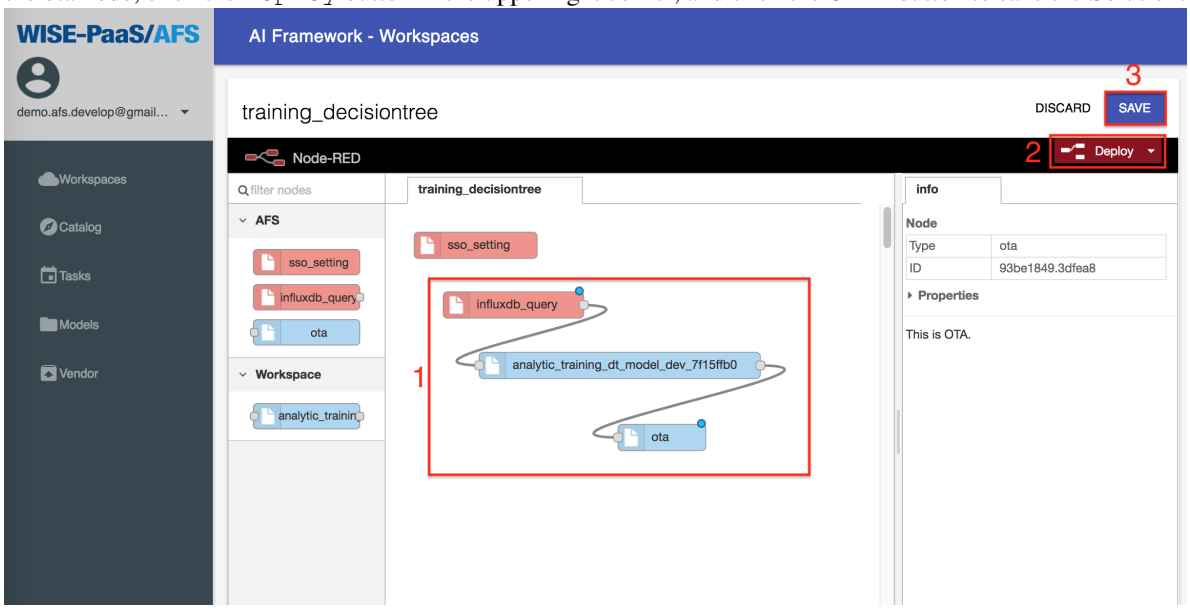
- Select Features: Select which fields are to be put into the model for training (can be multiple select). In the field, please select the fields KW_EQUIPMENT, KW_FAN, KW_SUMMARY, PRESSURE_OUTPUT, STATUS_FAN, VOLTAGE_INPUT, and EVENT.
- Select Numerical Features: Pick out the fields selected by select_features, which are the numeric fields (can be multiple selected but not fully selected, or not selected). Please select KW_EQUIPMENT, KW_FAN, KW_SUMMARY, PRESSURE_OUTPUT, STATUS_FAN, VOLTAGE_INPUT in the field.

- Select Target Feature: Select the target of training. Please select EVENT in the field.
- Map Column: The value of this field is the JSON Key value (can't be changed).

5. Pull the ota node from the list in the left side, and setup the parameters. Select the edge device and storage that were setuped in *Pre-condition*.

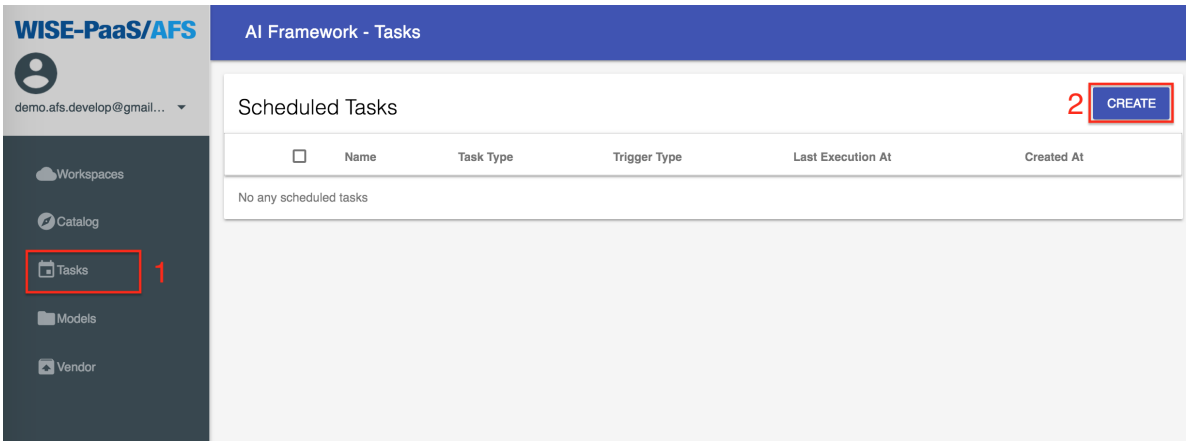


6. Connect the Influxdb_query node to the training_dt_model node, then connect the training_dt_model node to the ota node, click the `Deploy` button in the upper right corner, and click the `SAVE` button to save the Solution.

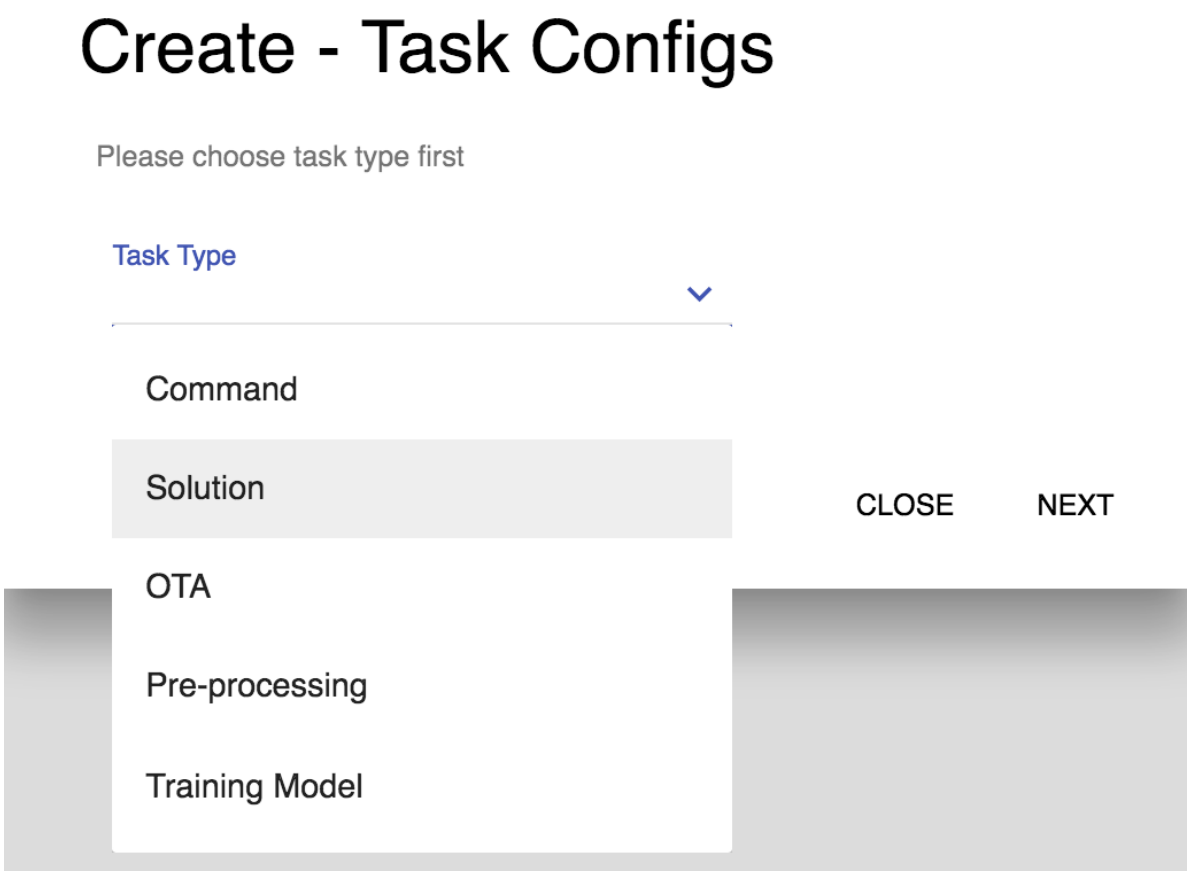


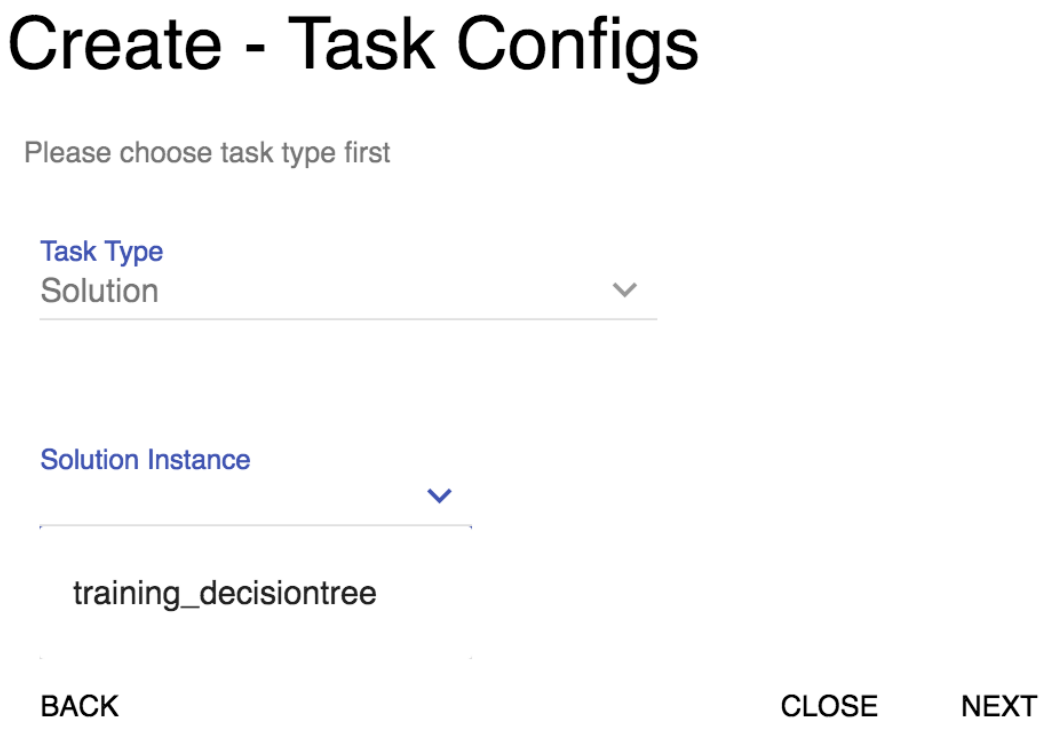
7. Create a new Solution Task.

   a. Click Tasks from the left menu, create a new task named training_decisiontree_task, and click the NEXT but-

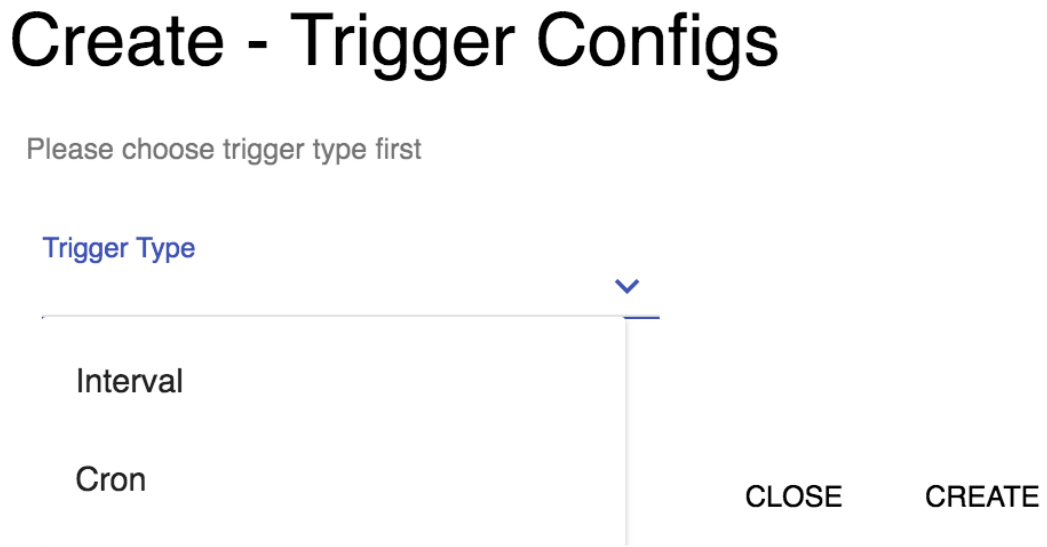
ton. 

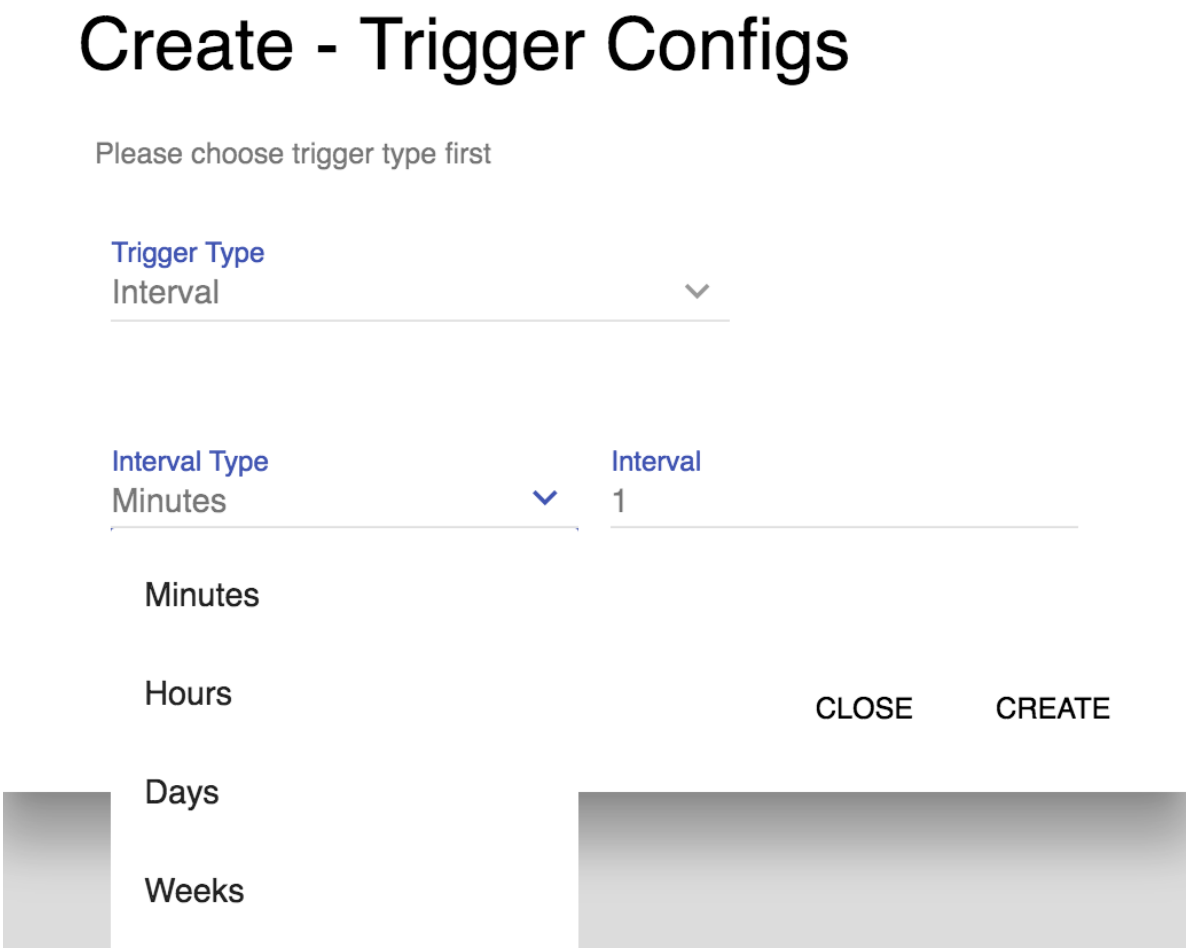
b. Select the **Solution** in Task Type, and select **training_decisiontree** Solution Instance. Then, click NEXT.


Create - Task Configs
Please choose task type first
Task Type
Command
Solution
OTA
Pre-processing
Training Model
CLOSE
NEXT

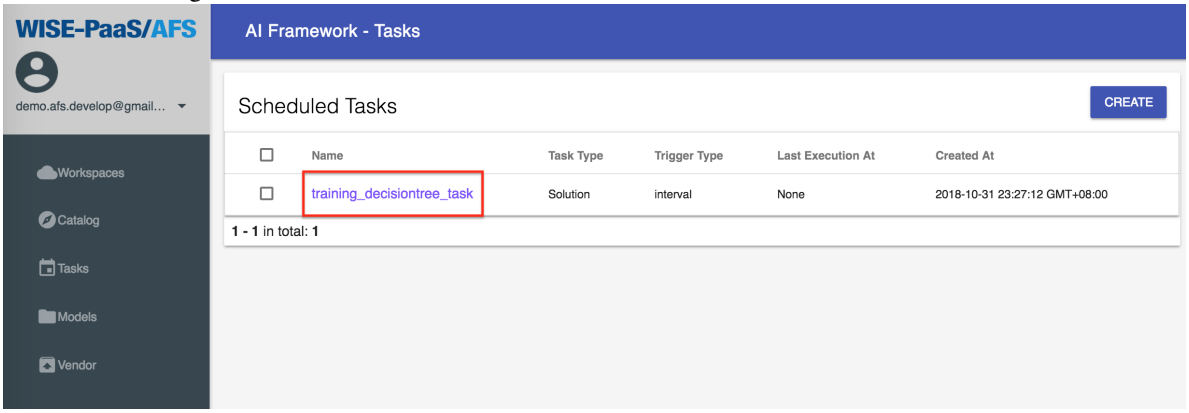


c. Select **Interval** in the Trigger Type, and selects **Minutes** in the Interval Type. The Interval fills in "1". Then, click CREATE.

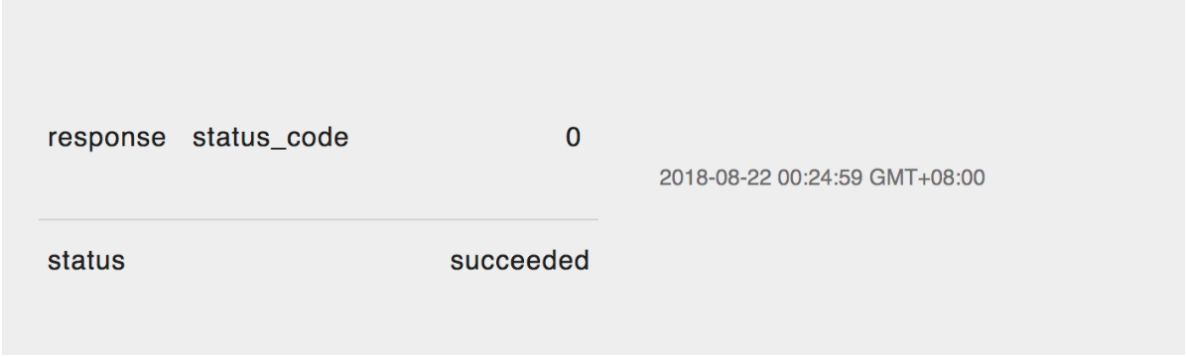




8. Click training_decisiontree_task to enter the task to see the results.



9. Wait a minute, the task will start executing. After the execution is successful, the status will be displayed as succeeded. If it does not appear after 1 minute, please press f5 to refresh the page.

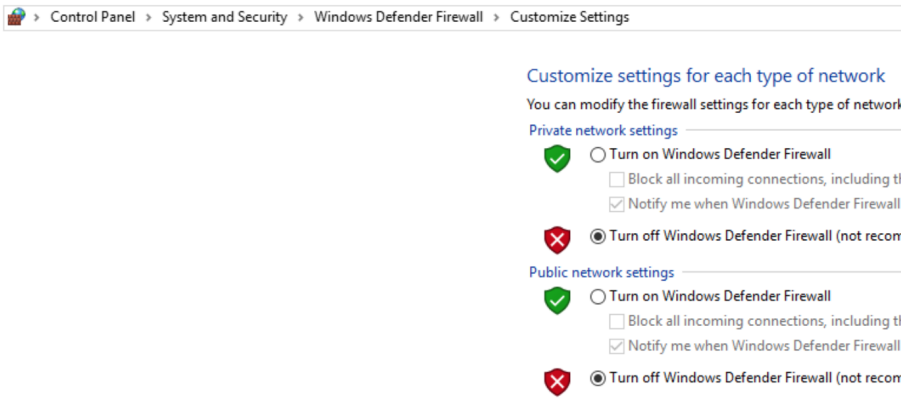

response status_code 0
2018-08-22 00:24:59 GMT+08:00
status succeeded

CHAPTER 16

SCENARIO 3. Inference Engine

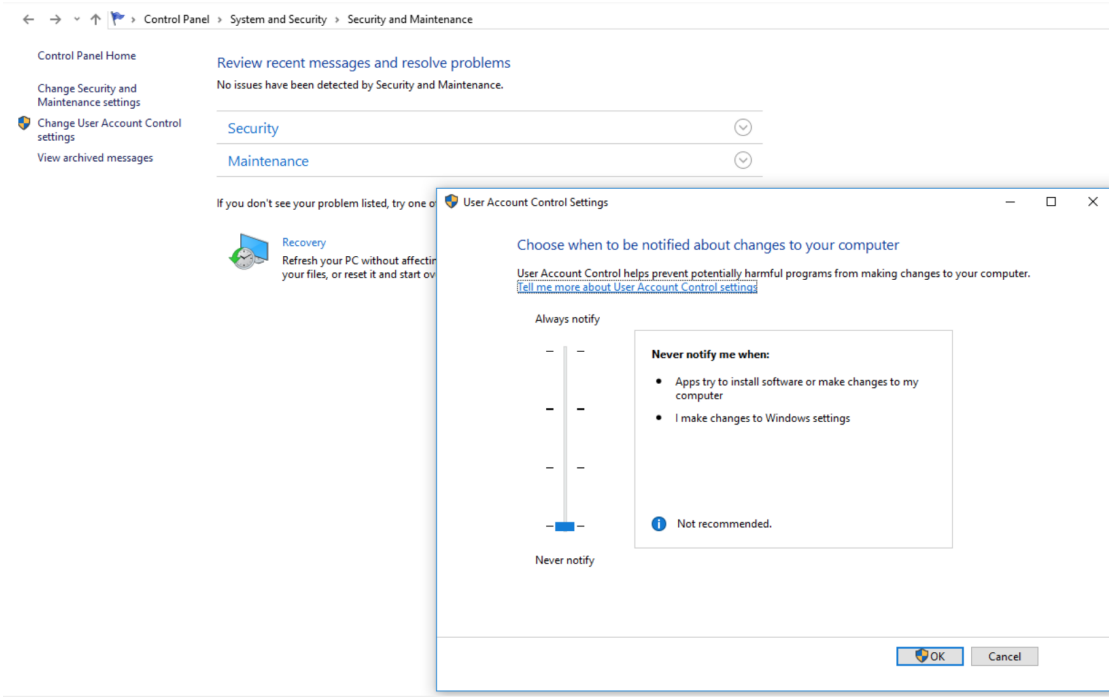
16.1 Pre-condition

- The OS of edge devices must be the **Windows 10 Pro** 64-bit version, and **Build 14393 or later**.
- The language of OS must be in **Simplified Chinese**, **Traditional Chinese**, and **English**.
- Turn on the Hyper-V in Windows 10. About the steps, please refer the document.
- The edge devices must be installed the **RMM Agent (v-1.0.16)**, and registed in RMM Server.
- Get the application of packaging (OTAPackager-1.0.5.exe). [Download]
- Download the files for package as follows:
  - Docker installer. [Download]
  - Three .bat files (include install_docker.bat, start_docker.bat, start_inference.bat). [Download]
  - SSL credential (registry.cert). [Download]
- Setup for login automatically after rebooting, please refer the page.
- Close the firewall.
  - Control Panel > System and Security > Windows Defender FireWall > Customize Settings.



  - Turn off Windows Defender Firewall.

- Close the notification.
  - Control Panel > System and Security > Security and Maintenance > Change User Account Control settings.



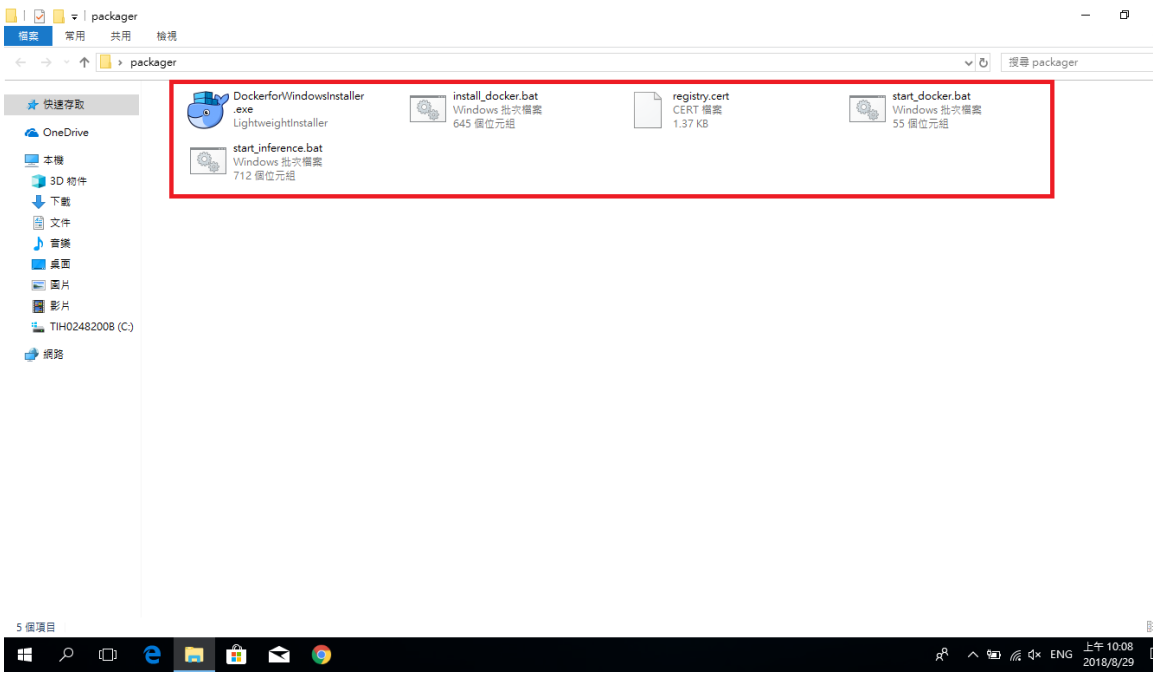
  - Set “Never notify”.

- The docker offical suggestion before installing, please refer the docker docs.
  - Windows 10 64bit: Pro, Enterprise or Education (1607 Anniversary Update, Build 14393 or later).
  - Virtualization is enabled in BIOS. Typically, virtualization is enabled by default. This is different from having Hyper-V enabled. For more detail see Virtualization must be enabled in Troubleshooting.
  - CPU SLAT-capable feature.
  - At least 4GB of RAM.

16.2 Start to Install Inference Engine

1. Use the OTApackager APP to package the required files.

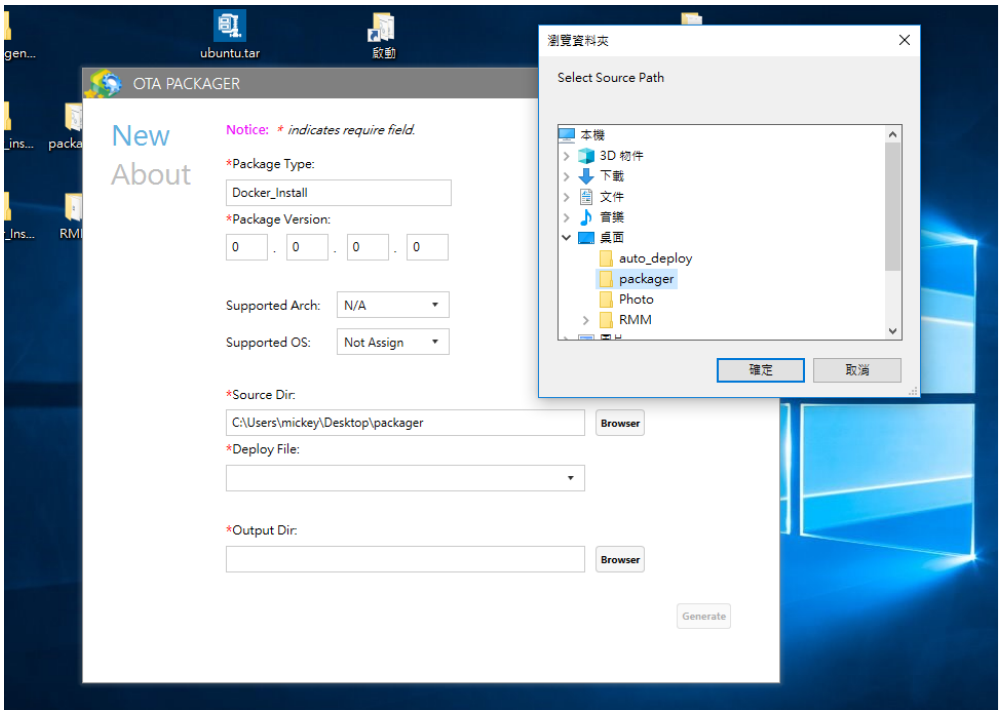
a. The required files.



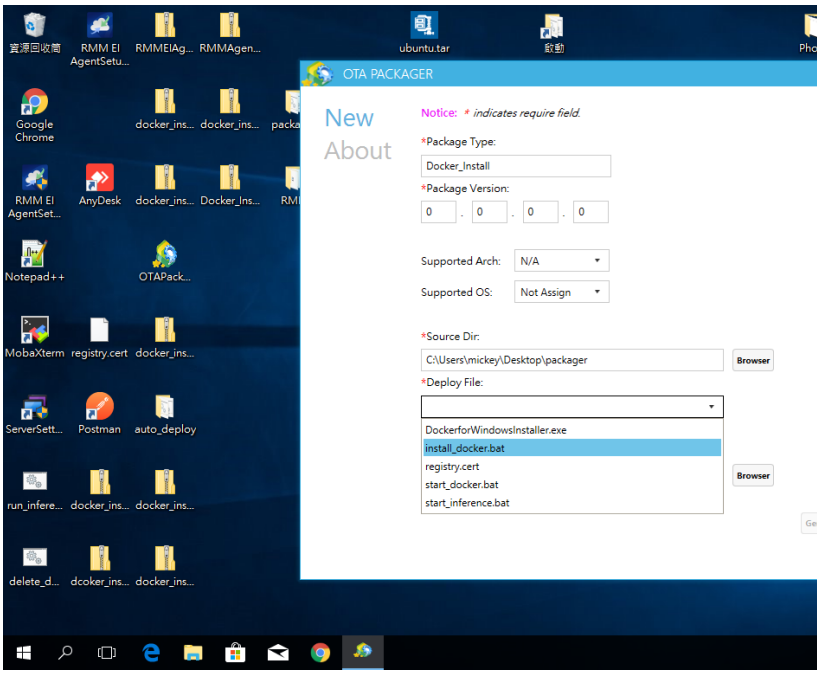
b. Edit "install_docker.bat", the file path should be modified to matching the path in the edge device.

```
copy /Y start_docker.bat "C:\Users\kai\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\start_docker.bat"
copy /Y start_inference.bat "C:\Users\kai\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\start_inference.bat"
certutil -addstore "TrustedPublisher" registry.cert
set file="C:\Program Files\Docker\Docker\Docker for Windows.exe"
if exist %file% (
  echo file is exists
)else (
  "Docker for Windows Installer.exe" install --quiet -Verb RunAs
   net localgroup docker-users kai /add
)
set docker_deamon="C:\Users\kai\.docker"
if not exist %docker_deamon% (
  md C:\Users\kai\.docker
)
(
echo { "registry-mirrors": [], "insecure-registries": [ "23.98.43.195:443" ], "debug": true, "experimental": false}
)>"C:\Users\kai\.docker\daemon.json"
DISM /Online /Enable-Feature /All /FeatureName:Microsoft-Hyper-V /Quiet
shutdown.exe /r /t 90
```

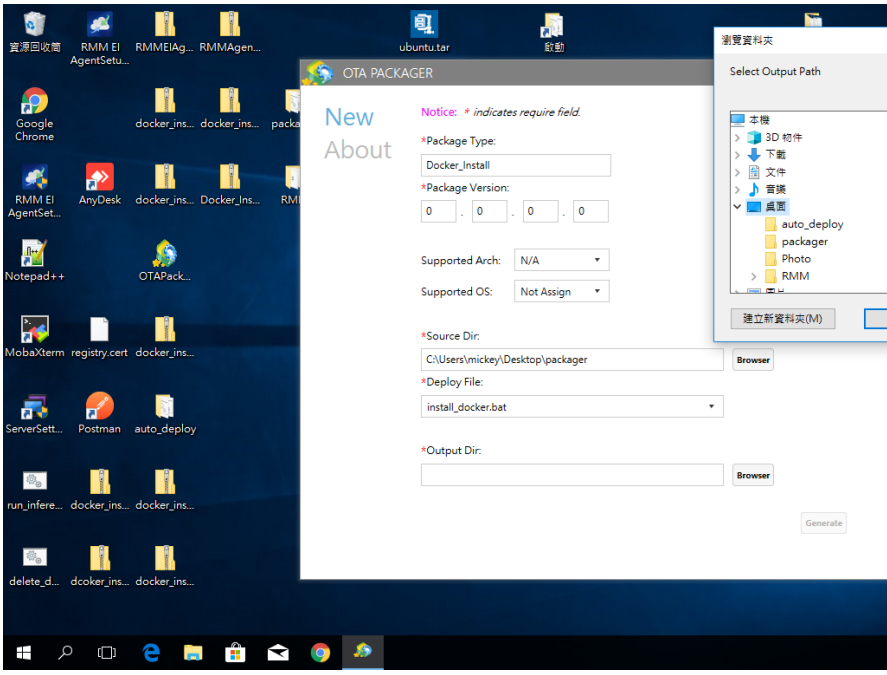
c. Enter the Package Type, Package Version, then select the path for saving the package file.



d. Select **install_docker.bat** to be the "Deploy File".

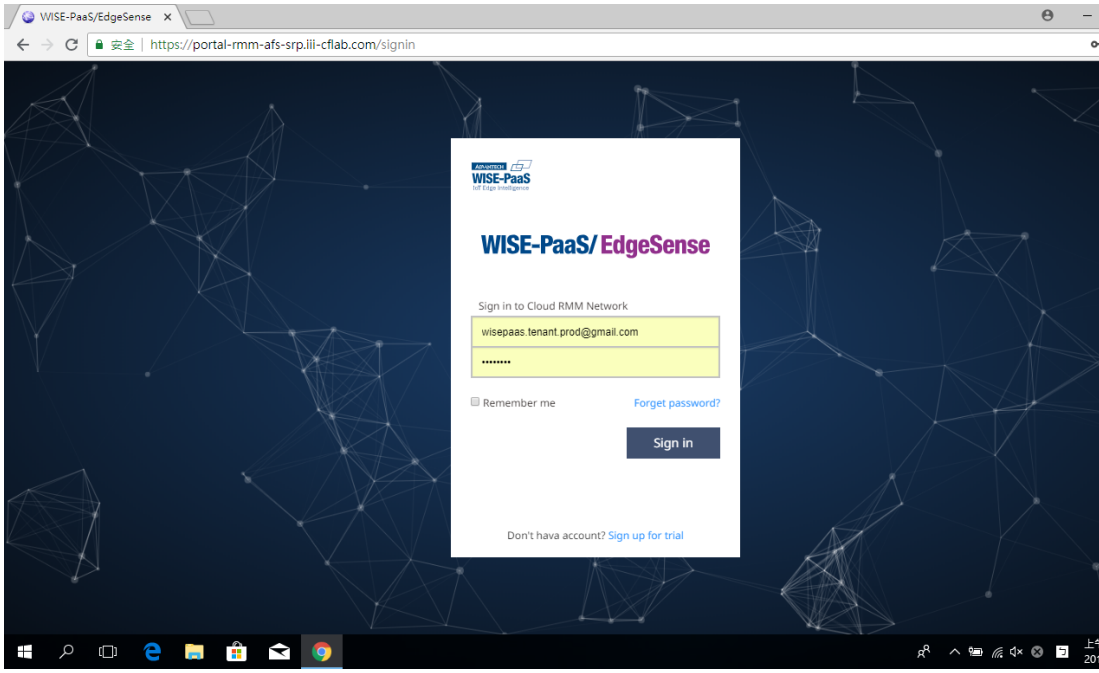


e. Select the folder for saving the package file.

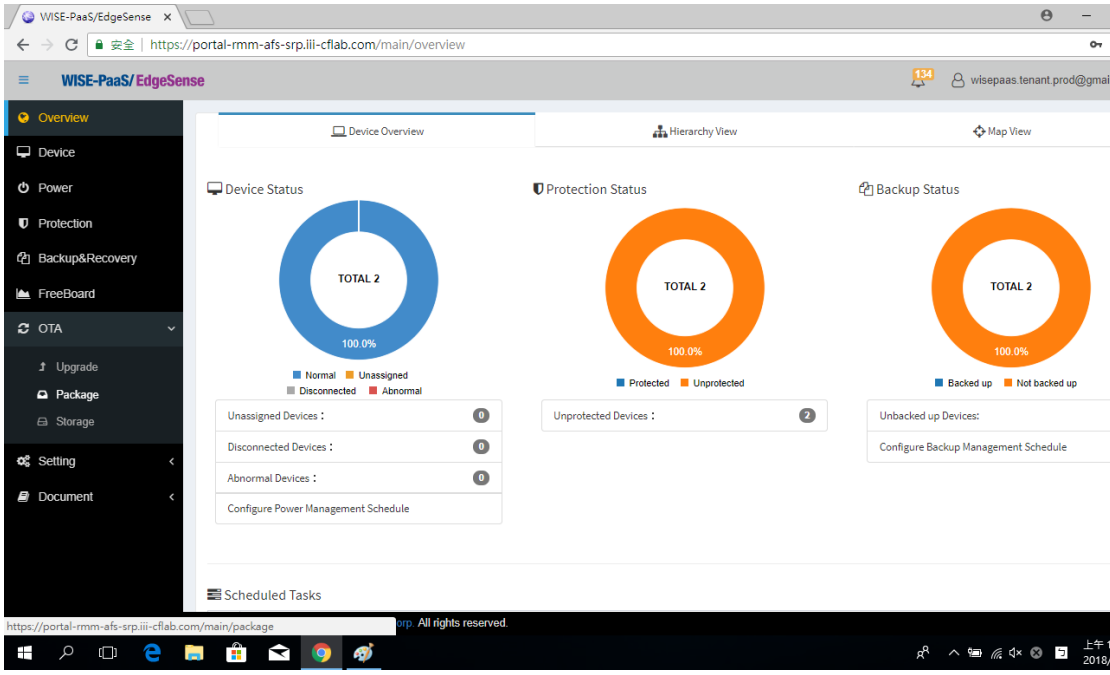


2. Login to **RMM Portal**, and upload the package file.

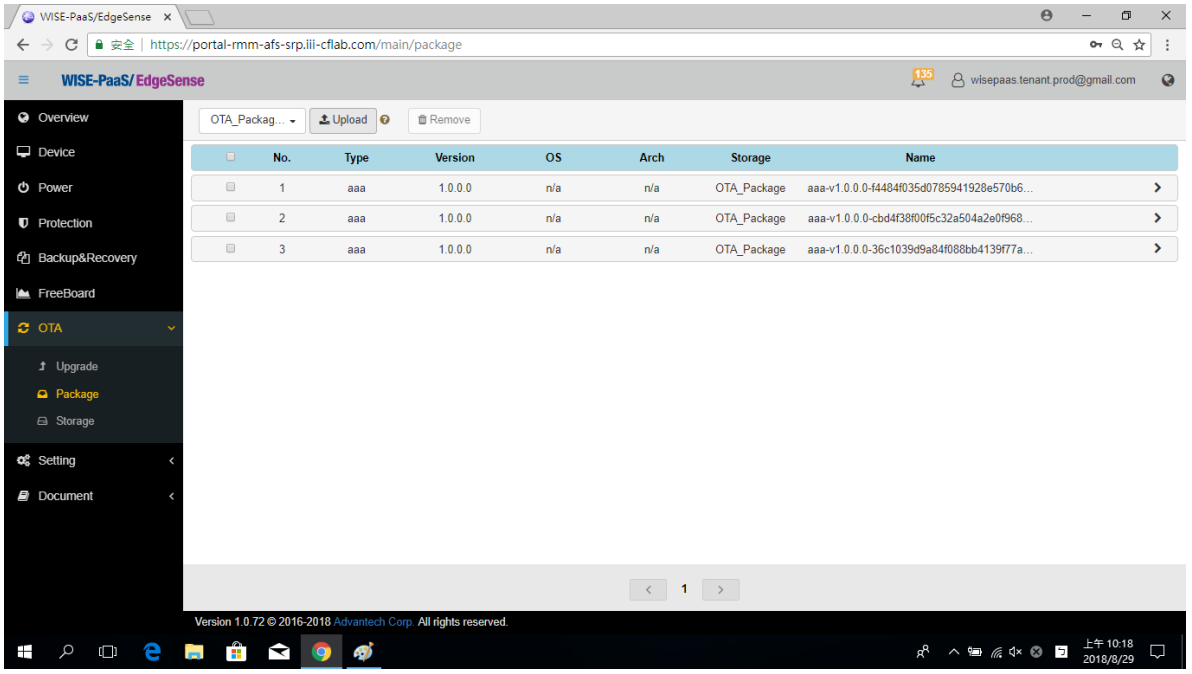
a. Login to **RMM Portal**.



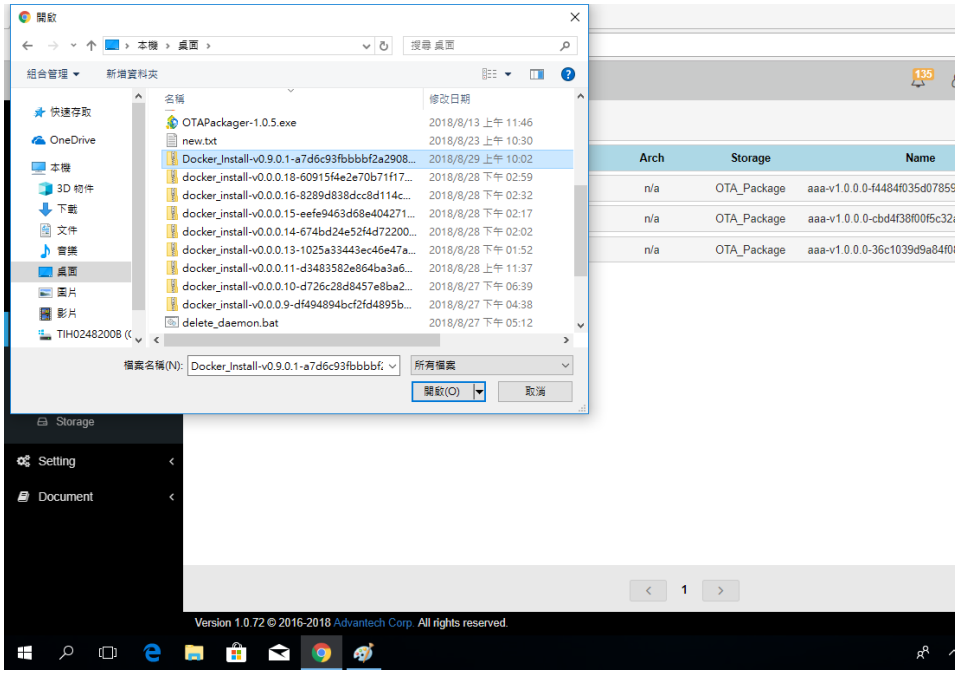
b. Click OTA > Package.



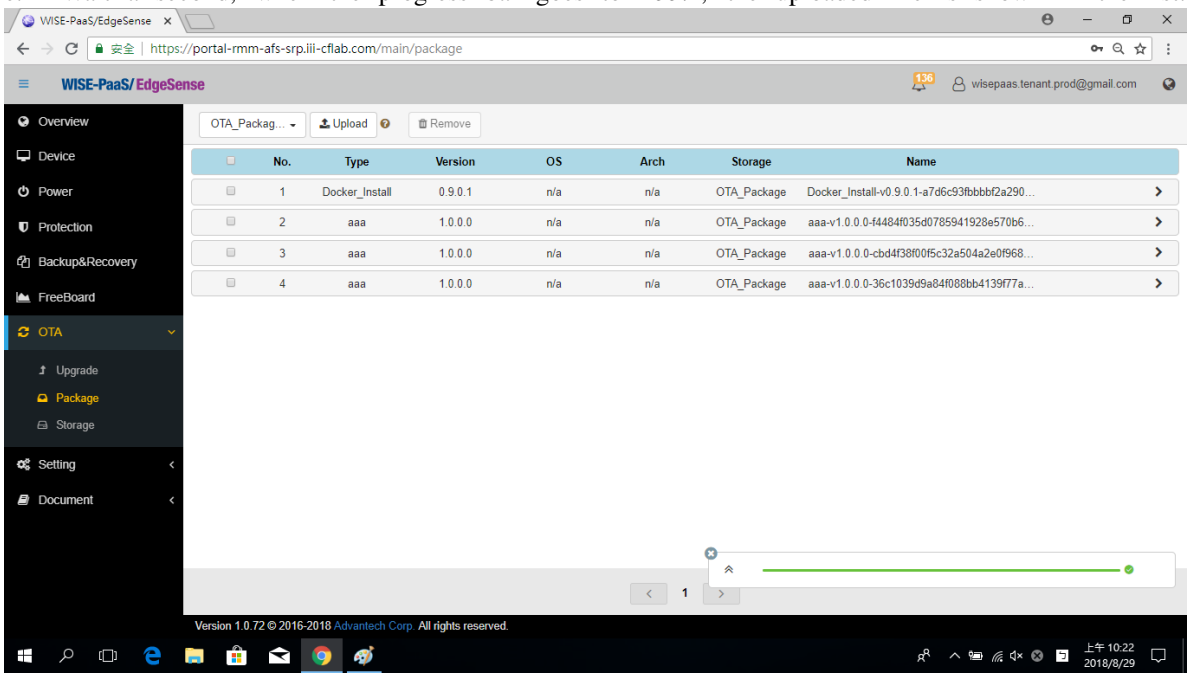
c. Click **Upload**.



d. Select the package file for uploading.

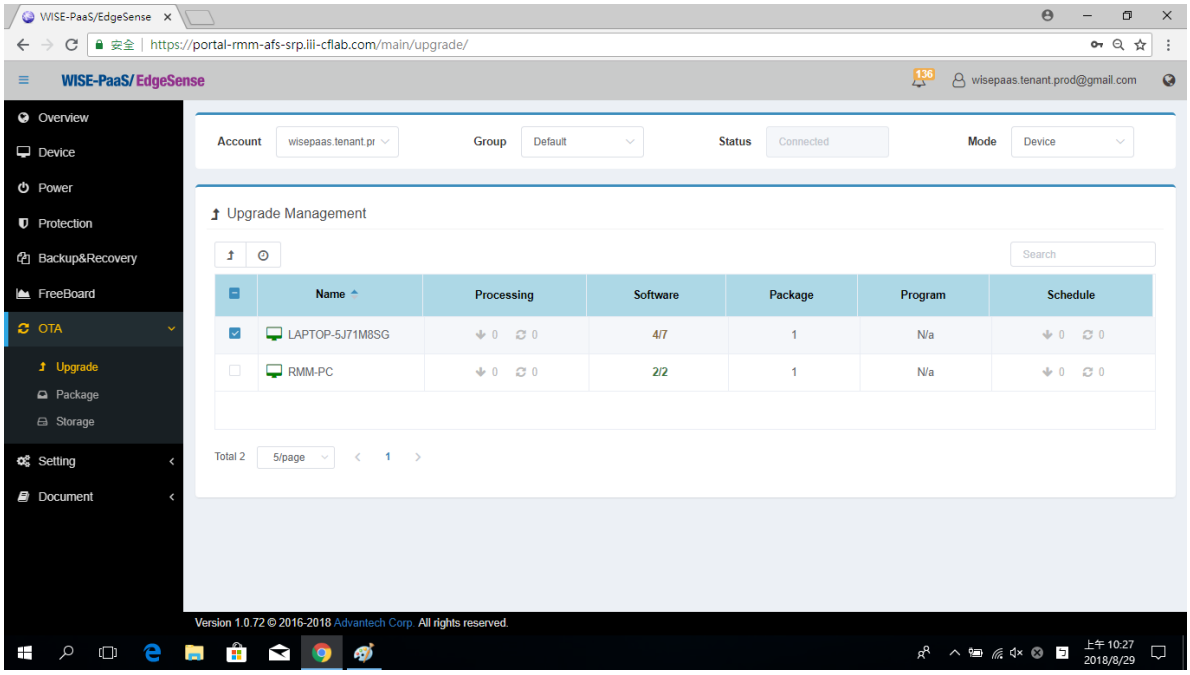


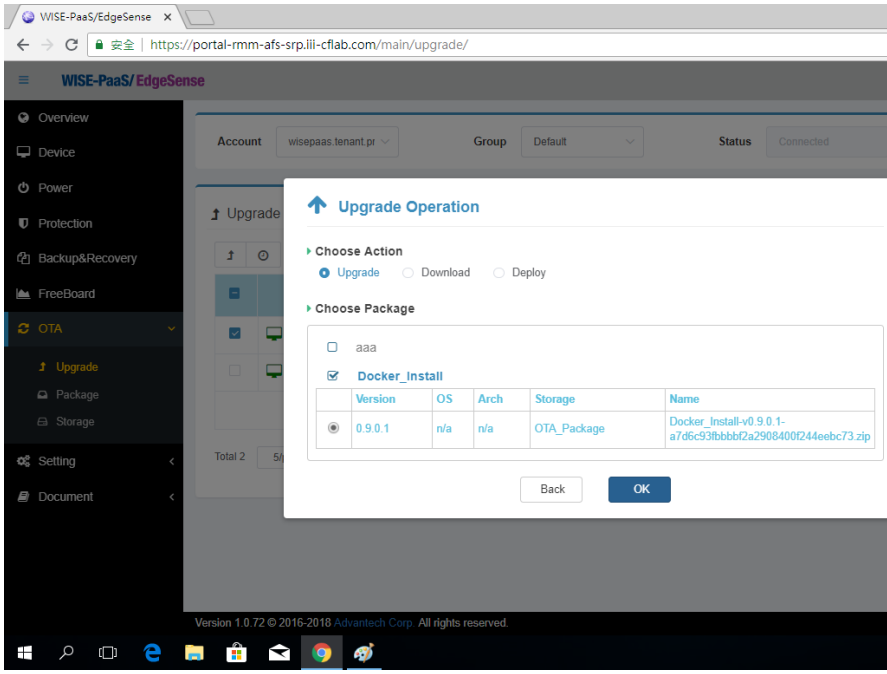
e. Wait a second, when the progress bar goes to 100%, the uploaded file is shown in the list.



3. Send the uploaded file to the edge device for installing automatically.

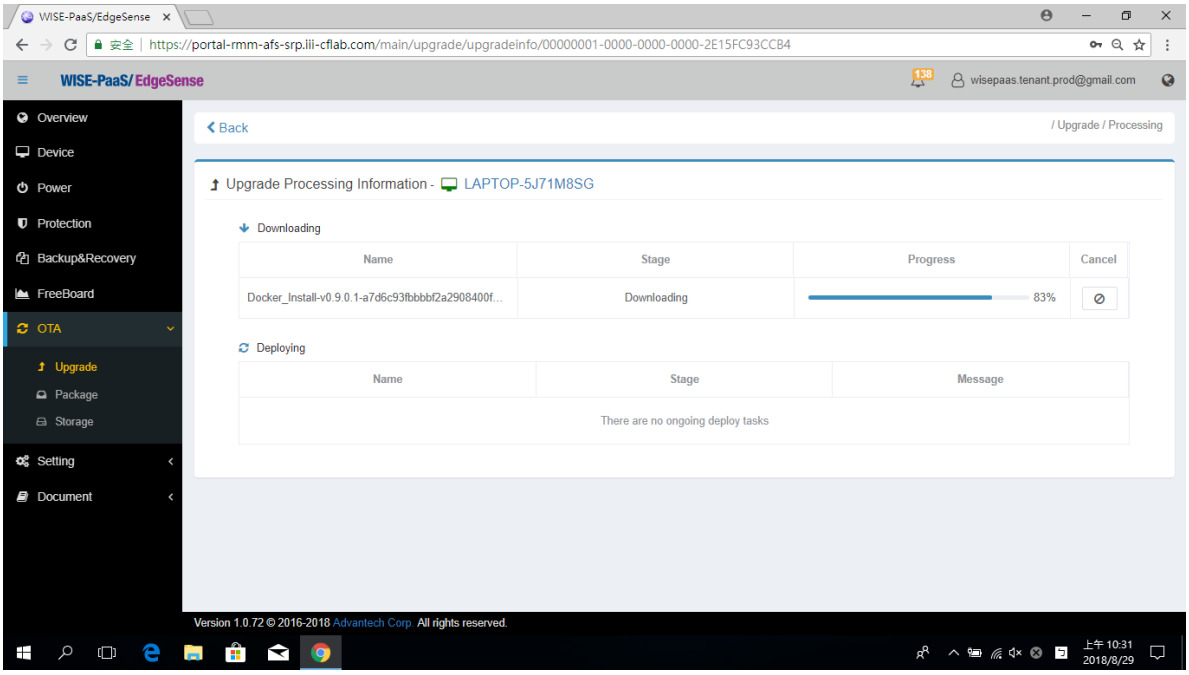
   a. Click OTA > Upgrade. Then, select the device to be installed.





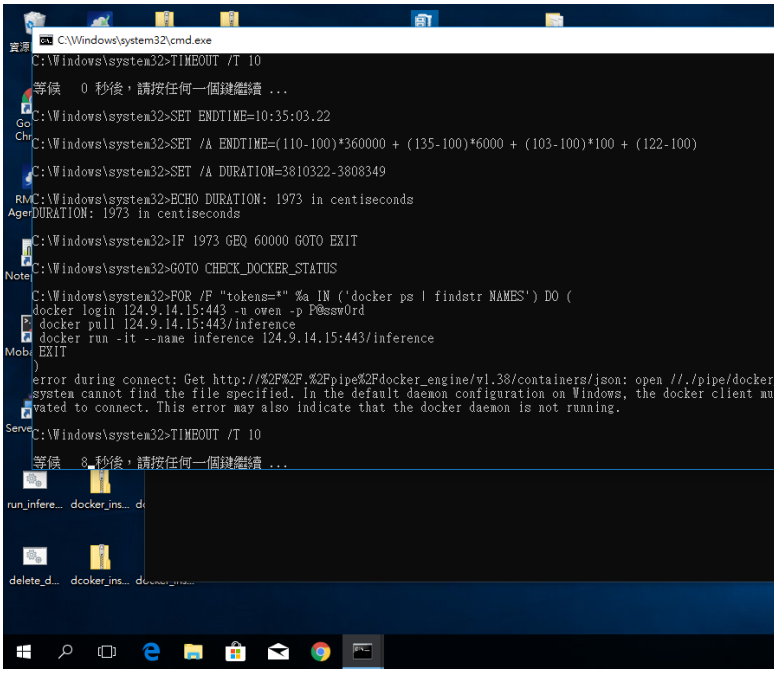
b. Selcet the package which want to **Upgrade**.

c. When the progressing bar goes to 100%, the edge device downloaded the package file completely, and start to



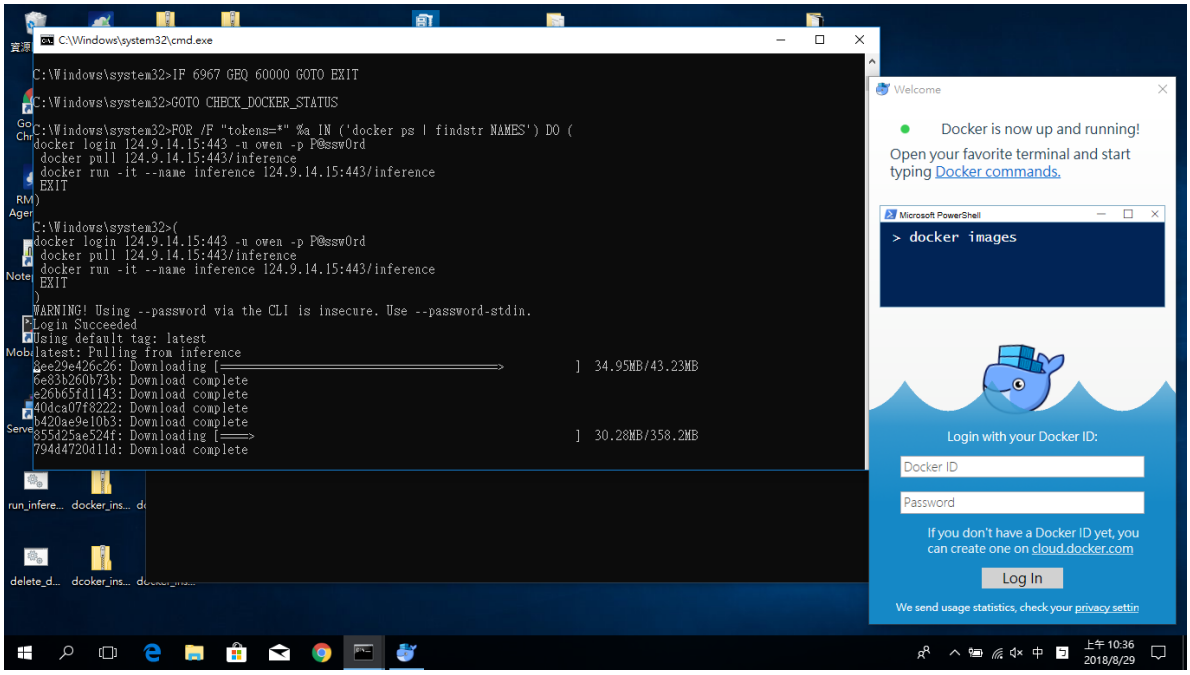
install it.

4. Before installing the package, the edge device restart once. The **Docker** in the edge device starts automatically, and the inference engine runs.



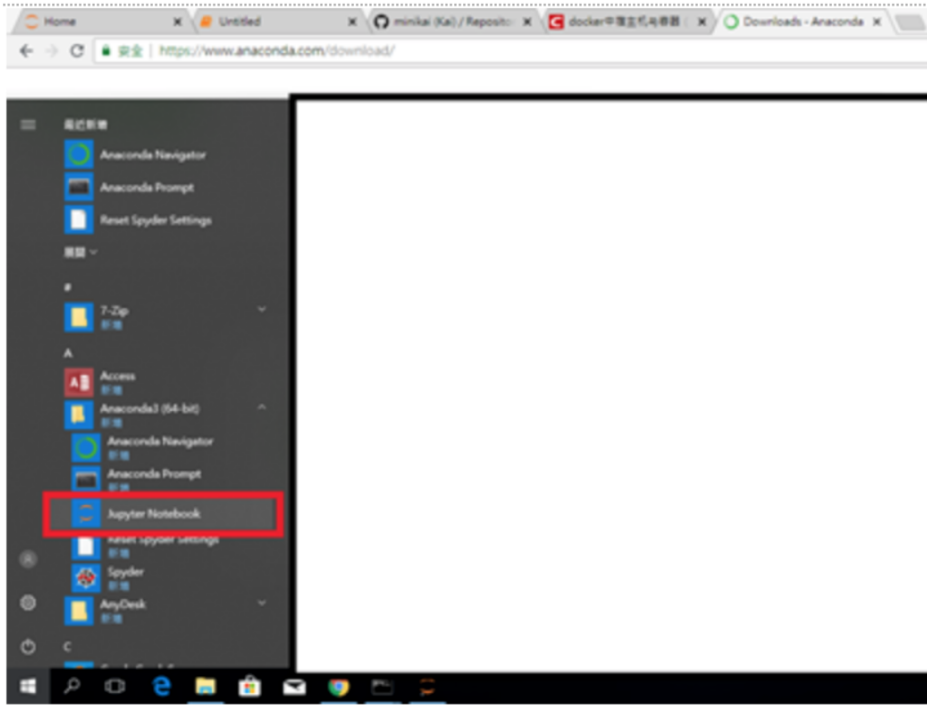
a. The screenshot shows when the installation is running.

b. In the screenshot, it shows the required images are downloading.



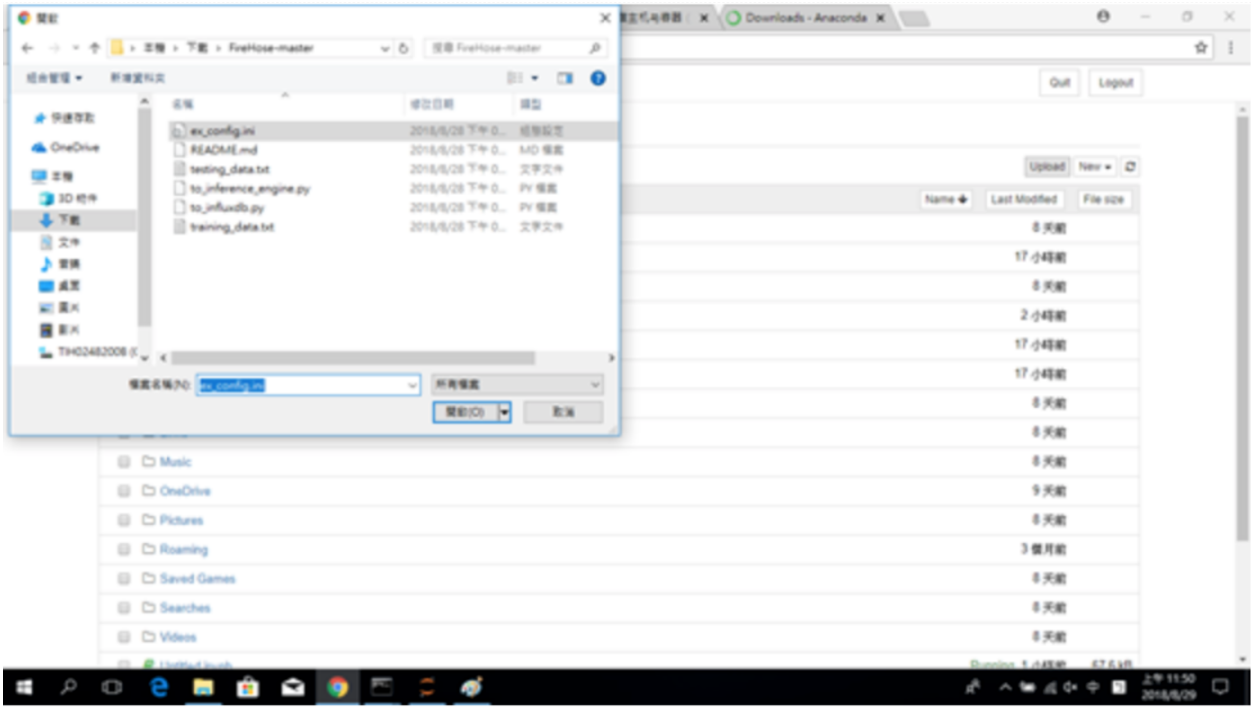
Finally, an edge device has been installed the inference engine automatically. Therefore, if there are many edge devices need to install the inference engine, we just need pick multiple devices in **Step 3.**, and they will be installed completely.

Now, we can use the model which is trained in Scenario 2. to inference.a. Confirm that the model is trained successfully in Scenario 2., and can be delivered to edge device by OTA.b. Download the anaconda (with python 3.6), and install it in the edge device. [Download]



c. Start the **Jupyter Notebook** from application.

d. Download the firehose for testing the inference engine.e. Click the Upload button at the top right to upload ex_config.ini, firehose.ipynb, and testing_data.csv to jupyter.



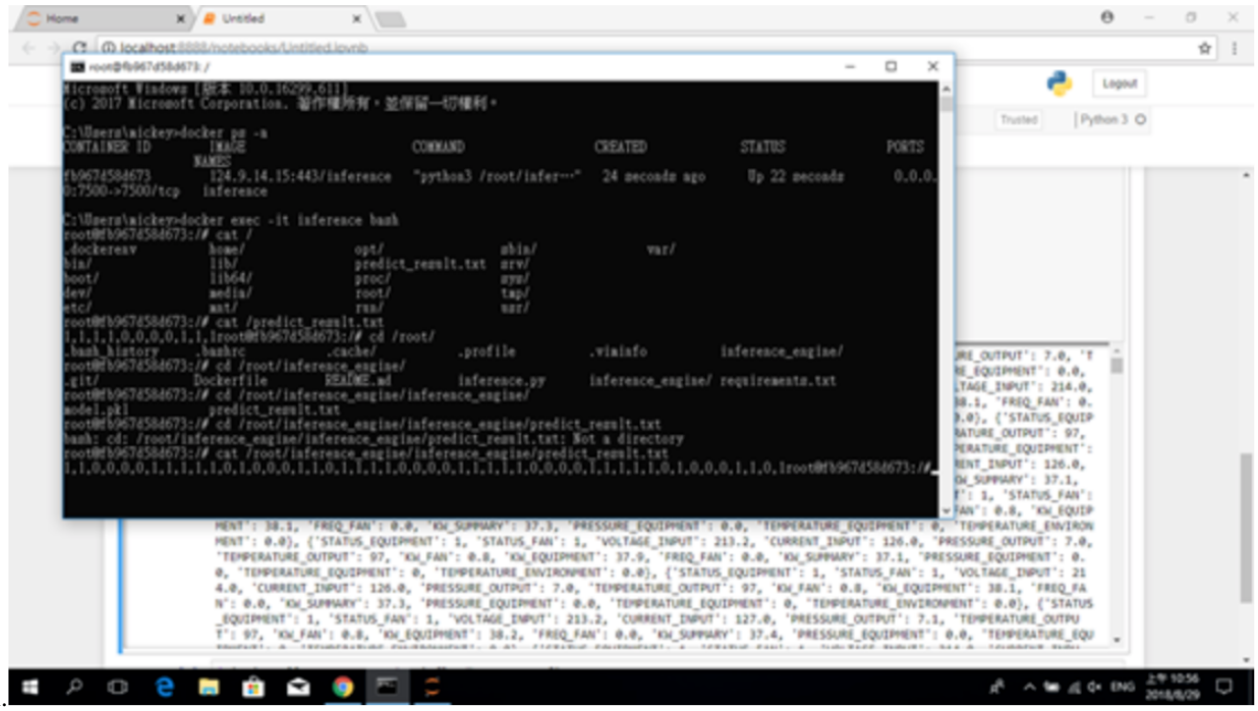
f. Click and modify ex_config.ini, and add “http://127.0.0.1:7500/predict” after “url=”. Then, save the file.

g. Open the firehose.ipynb just uploaded on jupyter and click `Run` to execute.

In [4]:
```
import requests
import json
import pandas as pd
import configparser
import csv
import datetime
import numpy as np
import time



config = configparser.ConfigParser()
config.read('ex_config.ini')
duration_API = config['push_data']['duration_API']
url = config['api']['url']
df1 = pd.read_csv('testing_data.csv')
df2 = df1[['STATUS_FAN','VOLTAGE_INPUT','PRESSURE_OUTPUT','KW_FAN','KW_EQUIPMENT','KW_SUMMARY']]
y = int(df2.shape[0])
i = 1

while i < y :
    row = df1[i:i+1]
    arr = row.values.tolist()
    data={}
    data['STATUS_FAN']=arr[0][1]
    data['VOLTAGE_INPUT']=arr[0][2]
    data['PRESSURE_OUTPUT']=arr[0][3]
    data['KW_FAN']=arr[0][4]
    data['KW_EQUIPMENT']=arr[0][5]
    data['KW_SUMMARY']=arr[0][6]
    data_list=[]
    data_list.append(data)
    json_data = {}
    json_data['data']=data_list
    r = requests.post(url, json=json_data)
    time.sleep(int(duration_API))
    print(json_data)
    i = i+1
```

```
{'data': [{'STATUS_FAN': 0.0, 'VOLTAGE_INPUT': 213.6, 'PRESSURE_OUTPUT': 6.25, 'KW_FAN': 0.15, 'KW_EQUIPME
NT': 25.48, 'KW_SUMMARY': 4.06}]}
{'data': [{'STATUS_FAN': 1.0, 'VOLTAGE_INPUT': 213.6, 'PRESSURE_OUTPUT': 1.85, 'KW_FAN': 0.4, 'KW_EQUIPMEN
```

h. Login to inference_engine, and see the prediction results.1. Execute `$ cmd` to open the command window.2. Execute `$ docker exec -it inference bash`.3. To check if the model is normally dispatched into the inference engine, we can execute `$ ls /root/inference_engine/inference_engine/` to check the model.pkl exists or not. (The model name must be “model.pkl”.)4. Execute `$ cat /root/inference_engine/inference_engine/predict_result.txt` to check if the predicted value continues to increase, if the representative is


Home
Untitled
localhost:8888/notebooks/Untitled.ipynb
Logout
Trusted
Python 3
Microsoft Windows [版本 10.0.16299.611]
(c) 2017 Microsoft Corporation. 著作權所有，並保留一切權利。
C:\Users\aickey>docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
fb967458d673 124.9.14.15:443/inference "python3 /root/infer…" 24 seconds ago Up 22 seconds 0.0.0.0:7500->7500/tcp inference
C:\Users\aickey>docker exec -it inference bash
root@fb967458d673:/# cat /
.dockerenv home/ opt/ sbin/ var/
bin/ lib/ predict_result.txt srv/
boot/ lib64/ proc/ sys/
dev/ media/ root/ tmp/
etc/ mnt/ run/ usr/
root@fb967458d673:/# cat /predict_result.txt
1,1,1,1,0,0,0,0,1,1,1root@fb967458d673:/# cd /root/
.bash_history .bashrc .cache/ .profile .viminfo inference_engine/
root@fb967458d673:/# cd /root/inference_engine/
.git/ Dockerfile README.md inference.py inference_engine/ requirements.txt
root@fb967458d673:/# cd /root/inference_engine/inference_engine/
model.pkl predict_result.txt
root@fb967458d673:/# cd /root/inference_engine/inference_engine/predict_result.txt
bash: cd: /root/inference_engine/inference_engine/predict_result.txt: Not a directory
root@fb967458d673:/# cat /root/inference_engine/inference_engine/predict_result.txt
1,1,0,0,0,0,1,1,1,1,1,1,0,1,0,0,0,1,1,0,1,1,1,1,0,0,0,0,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,0,1,0,0,0,1,1,0,1root@fb967458d673:/#

successful.

CHAPTER 17

SCENARIO 4. AFS Vender

The development process can be done offline through Vendor.

1. The module can be installed offline through Vendor. About more details, please refer to documents.
2. About how to manage the Vendor, including module upload, download, and delete the package, please refer to documents.

CHAPTER 18

SCENARIO 5. AFS Tasks

18.1 Create a task

The detailed steps are included in Scenario 2., please refer to the steps 7 to 9 of Scenario 2.

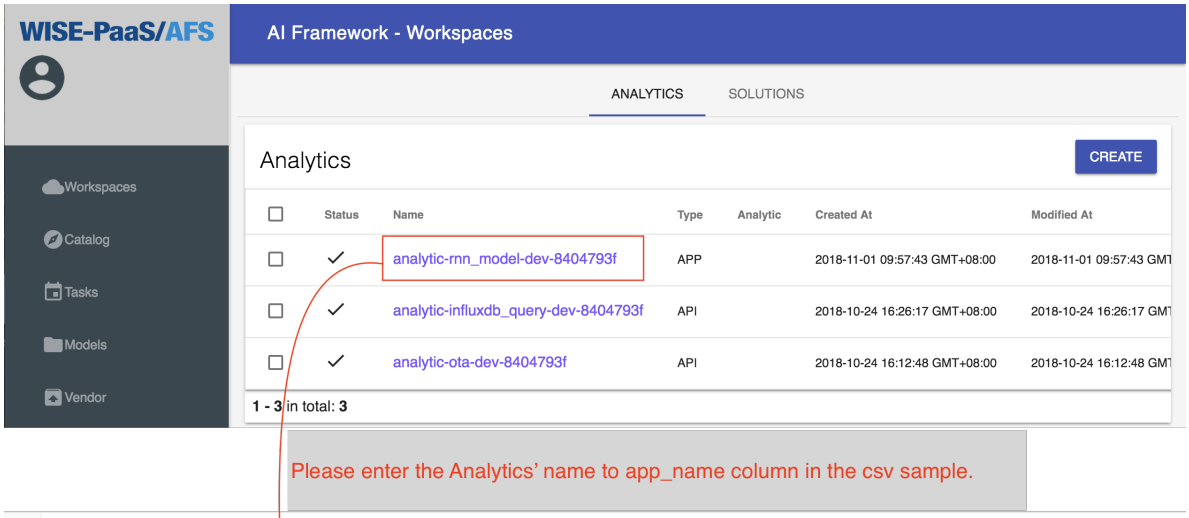
18.2 Create multiple tasks

1. Download “multiple_task_example.csv” to make the list of tasks.

   a. Click the CREATE button in the upper right corner of the Tasks page, click the button in the upper right corner of the pop-up window and click **Create multiple tasks**.

   b. Click the link to download csv example.

   c. Copy the sample to a text editor and name the file multi_task.csv. Please enter the Analytics’ name to app_name column in the csv sample.




```
name,task_type,app_name,command,arguments,trigger_type,minutes,hours,cron,device,storage,model_repository,username,password,solution_instance_name
training_rnn_cron,command,analytic-rnn_model-dev-8404793f,run_jnb rnn_model-dev.ipynb -m false,,cron,,,* * * * *,,,,,,
training_rnn_interval,command,analytic-rnn_model-dev-8404793f,run_jnb rnn_model-dev.ipynb -m false,,interval,1,,,,,,,,
training_dt_task,solution,,,,interval,1,,,,,,,,training_decisiontree
```

2. Select the csv file (please select the csv file created by 1.c above) and click `CREATE` to create the tasks.

CHAPTER 19

SCENARIO 6. AFS Model

AFS Model shows the performance of the model training. It has introduced in Step 8. to Step 9. of SCENARIO 1.

CHAPTER 20

Side Effect of Removing the Hidden Space

Before AFS v1.2.26, there was a space that storaged the Jupyter and Node-RED two applications. The information of the space wasn't shown in the Management Portal, so called "hidden space". Therefore, users can't know the quota of resource had been used. In order to declare the resource quota, the hidden space has been removed. All of apps are moved to the user's space, and they are listed in the "Application List", currently.

- To avoid the duplicated apps' name, the naming rules are revised as follows:
  - Jupyter => code-ide-{instance[0:8]}, {instance_id}-jupyter.{domain} => code-ide-{instance[0:8]}.{domain}
  - Node-RED => flow-ide-{instance[0:8]}, {instance_id}-node-red.{domain} => flow-ide-{instance[0:8]}.{domain}
  - {name}-dev => analytic-{name}-dev-{instance[0:8]}, {instance_id}-{name}-dev.{domain} => analytic-{name}-dev-{instance[0:8]}.{domain}
- After removing the hidden space, the apps are listed in the Management Portal. The users can remove any apps by themselves, but removing some specific apps will damage the AFS service instance. The list of apps as follows are the dependency of AFS, please DON'T remove them:
  - code-ide-xxxxxxxx
  - flow-ide-xxxxxxxx
- Currently, the users can remove the apps in the Management Portal. After removing the apps will cause that the status of apps can't be shown correctly in the AFS Portal.
- In the Online Code IDE, the existing notebooks can not be saved after editing.
  - Because of the AFS service instance used by the notebook to be edited is subscribed before removing the hidden space (before AFS v1.2.26 version). After removing the hidden space, the name and url of the online code IDE apps which are added by the users are changed (e.g., the name of apps is modified by "analytic-{name}-dev-{instance[0:8]}"). In the previous version of the service instance, when users edit the notebook, the corresponding name and url would not be found for SAVE.
  - Users are supposed to re-subscribe a new AFS service instance and move the existing notebooks to it.

CHAPTER 21

Troubleshooting

In the section, we provide some problems that users may encounter, and the solutions for reference.

21.1 Jupyter Kernel Die

1. Memory GC issue There are 2GB memory for each Jupyter notebook. It may occur the kernel restart when use too more memory. The example for releasing the memory is as follows.Before: When the API is called, it will occupy 512MB of memory.

    - GET /test

```
memory_str = ' ' * 512000000 * 1
print("OK")
```

    After: When the result is returne, the variables are deleted, and the memory will be released.

    - GET /test

```
memory_str = ' ' * 512000000 * 1
del memory_str
print("OK")
```

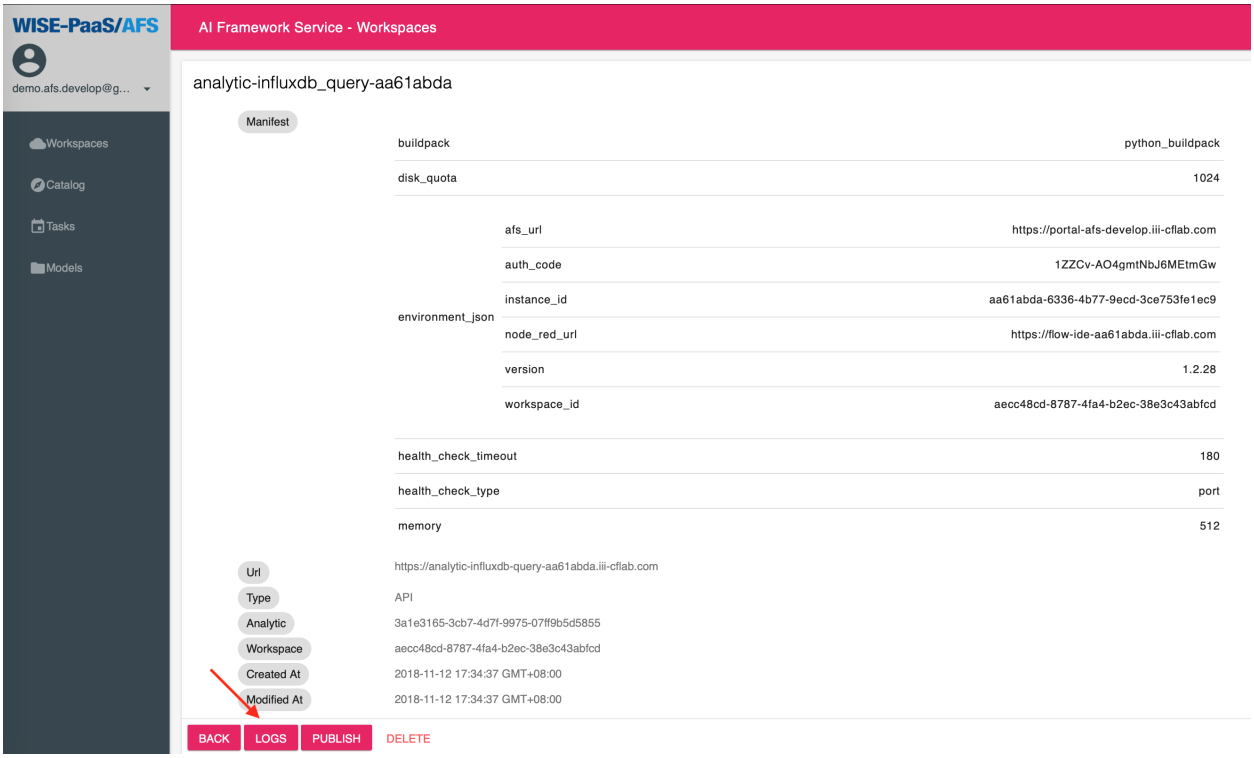
2. Disk full issue There are 2GB disk space for each Jupyter notebook, and there are about 1.2GB used for installing the Jupyter and related packages.
3. Dependency packages There are some dependency packages of Jupyter. They cause the kernel error when they have bug occasionally.

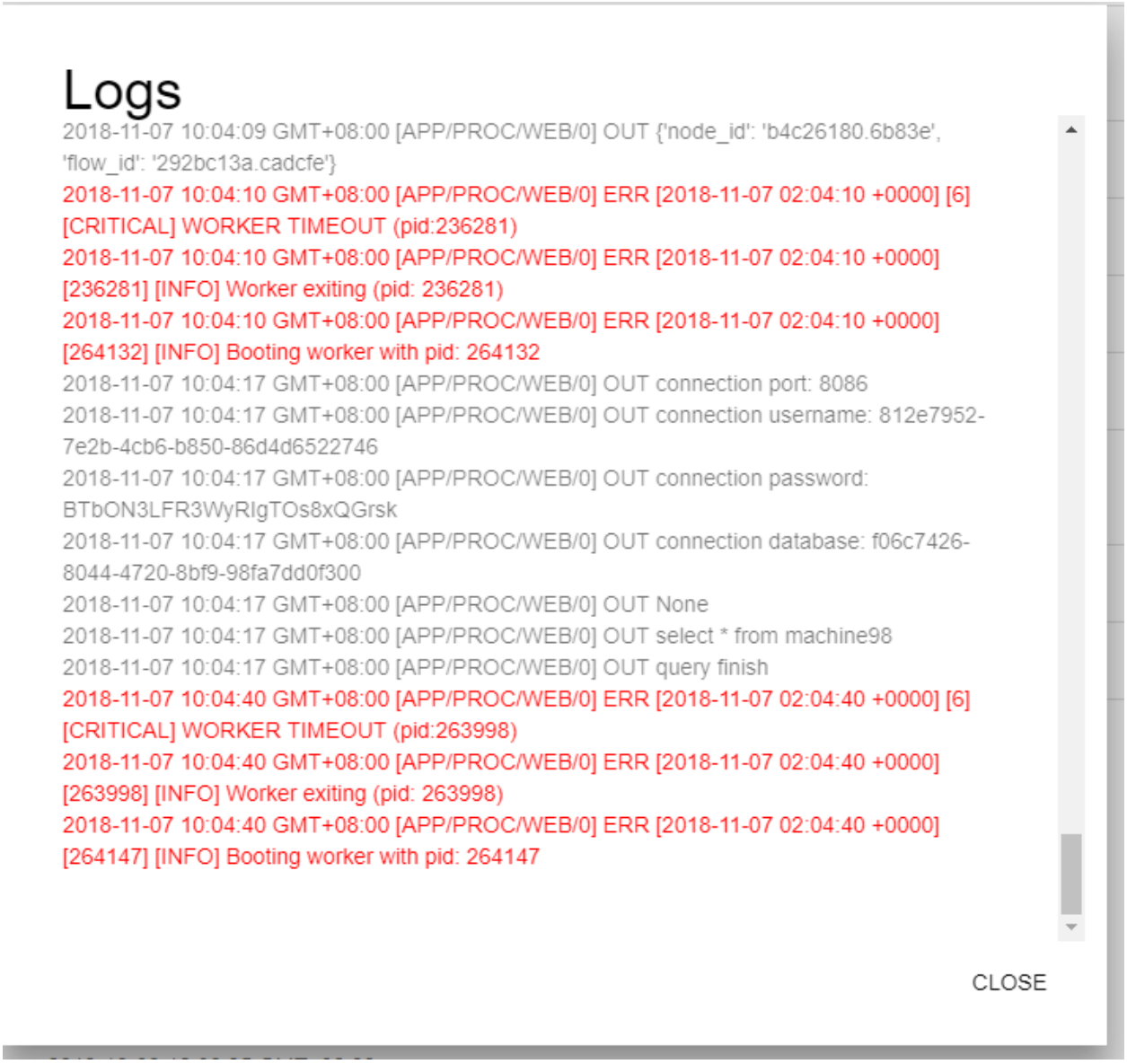
    - ipykernel
    - ipython
    - jupyter_client
    - jupyter_core

- traitlets
- ipython_genutils

21.2 Task Failed

The analytics and solutions can be scduled to execute automatically by **Tasks**. About the operations are introduced in the Tasks. However, there is limitation when the task works, and it's described in the section.The troubleshooting of task failed is introduced. When the problem occurs, we can check the log in the analytics. The example and steps are as follows: Please click the **Workspaces**, and click the analytic which want to check. Then, we can see the `LOGS` button, and click it. The logs are shown in the diagram. The message shows "WORKER TIMEOUT" that why the task failed. User can restart the app in the Management Portal, and create a new task for the analytic.





21.3 Other Issue

- When uploading the file which is less than 2GB, but the error occured, the error message: "StorageDataError: BotoClientError: Out of space for destination file."
  - **Root cause:** Checking the Jupyter in afs service instance and find that the disk is almost full, already used about 1.9GB. It causes an exception message when the Boto client is used to get file from the Blob store.
  - **Solution:** When subscribing the AFS service instance after version 1.2.26, the Jupyter and Node-RED would be deployed to the current AFS Instance. Users can use the CF CLI to obtain the current usage of the disk. (Management Portal only displays the size of the App, but can't display usage). If there is not enough space, the users can delete the application or restart the app by the CLI command.

  There are the commands to check the disk quota:

- Check the disk quota that the current APP are used: `cf app APP_NAME`
- Login to the APP: `cf ssh APP_NAME`
- Restart the App: `cf restart APP_NAME`

CHAPTER 22

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index