Event-Driven Smart Agriculture Monitoring Using AWS Cloud Services

MSc Research Project
MSc in Cloud Computing

VIVEKANANDA KANUKULA
Student ID: X23340312

School of Computing
National College of Ireland

Supervisor: **Jorge Mario Cortes Mendoza**

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	VIVEKANANDA KANUKULA
Student ID:	X23340312
Programme:	MSc in Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Jorge Mario Cortes Mendoza
Submission Due Date:	11/08/2025
Project Title:	Event-Driven Smart Agriculture Monitoring Using AWS Cloud Services
Word Count:	1303
Page Count:	13

Signature:	vivekananda
Date:	11th August 2025

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Event-Driven Smart Agriculture Monitoring Using AWS Cloud Services

VIVEKANANDA KANUKULA
X23340312

1 Introduction

This part describes the software, hardware, and cloud service-based needs of the implementation of the Event-Driven Smart Agriculture Monitoring Using AWS Cloud Services system. Its arrangement will provide a seamless mix of IoT devices, machine learning forecasts, and real-time dashboards with the help of AWS managed services and Pythonbased developer tools. It also lists the environment variables and python libraries needed to install, deploy, and sustain the project.

2 System Configuration

2.1 Software Requirements

- python : 3.10+
- Dashboard frame work: Streamlit v1.33+
- Development Environment: AWS Cloudshell, Jupyter Notebook (SageMaker Studio)
- MQTT Library:paho-mqtt v1.6+
- GitHub Actions
- Data Processing Libraries:pandas v2.2+, numpy v1.26+
- Learning Library Machine: scikit-learn v1.4+
- AWS CLI / SDK AWS CLI v2, boto3 v1.34+

2.2 Hardware/Hosting

- Local: laptop with ≥ 8GB RAM
- EC2 Instance (Dashboard): t3.micro or t3.small

3 AWS Services and Roles

The following AWS services were used in the project, along with their roles and key configurations:

- AWS IoT Core — Secure MQTT message broker to ingest sensor data. Key configurations: Create Thing, certificates, and policy; IoT Rule to trigger Lambda.
- AWS Lambda — Serverless processing that integrates IoT, DynamoDB, S3, SageMaker, and SNS. Key configurations: Python 3.10 runtime; least-privilege IAM role; environment variables set.
- Amazon SageMaker — Hosts the ML model (RandomForestClassifier) to predict irrigation. Key configurations: Model file (.joblib) and inference script (predict.py); endpoint instance type ml.t3.medium.
- Amazon DynamoDB — Stores recent sensor data and predictions for real-time access. Key configurations: Partition key device_id; sort key timestamp.
- Amazon S3 — Archives historical sensor and prediction data. Key configurations: Lifecycle policy to migrate older records to lower-cost storage.
- Amazon SNS — Sends SMS notifications when irrigation is needed. Key configurations: Create topic; phone number subscriptions.
- Amazon EC2 — Hosts the real-time Streamlit dashboard. Key configurations: Amazon Linux; security group open port 8501.

4 Local Environment Setup

1. AWS Account Create and Sign In

   Registered an aws account with https://aws.amazon.com with an email.
2. Network setup Multi-Factor Authentication (MFA) log-on safeguarding.
3. Log in to aws management console.
4. Access CloudShell AWS
5. Launched AWS CloudShell console by going through AWS Console to execute AWS Command line Interface without having to install any locally.

   confirmed CLI version of AWS aws -v.
6. Install Configure Dependencies on CloudShell
7. Installed python libraries (boto3, paho-mqtt, pandas, numpy, plotly, streamlit).
8. Set access keys on AWS CLI with aws configure.
9. AWS IoT Core Set up

   agriculture-IoT-Room device was a new IoT Thing (virtual device) created in AWS IoT Core.

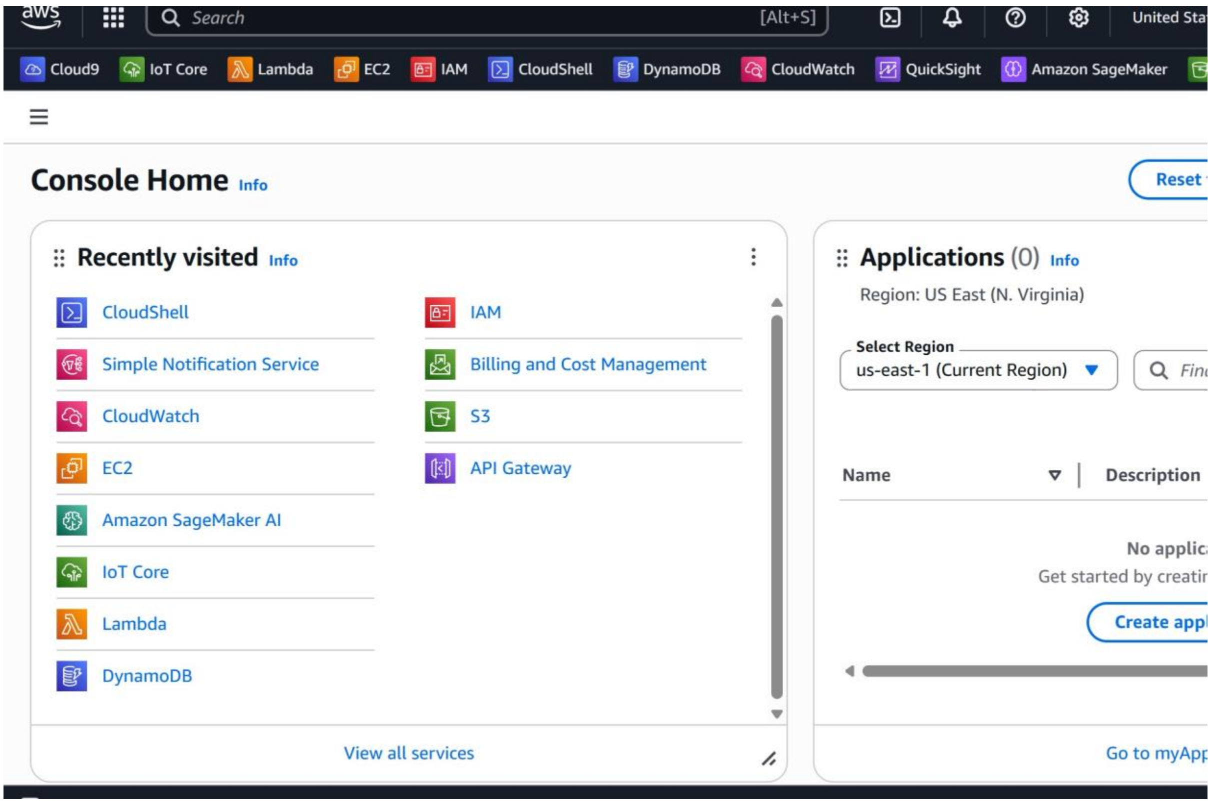
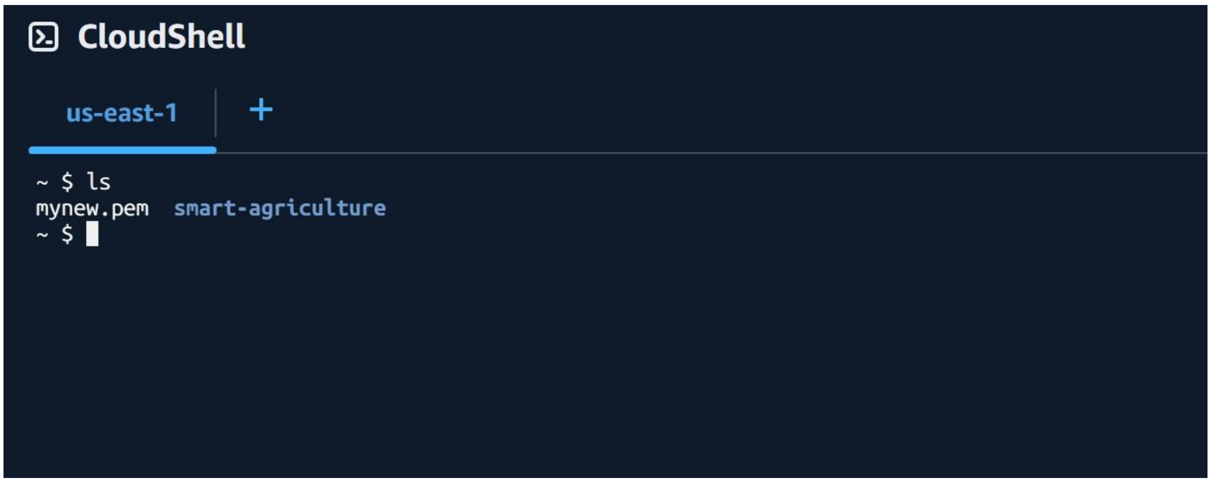


Figure 1: AWS Management Console



```
~ $ ls
mynew.pem  smart-agriculture
~ $ 
```

Figure 2: AWS Management Console

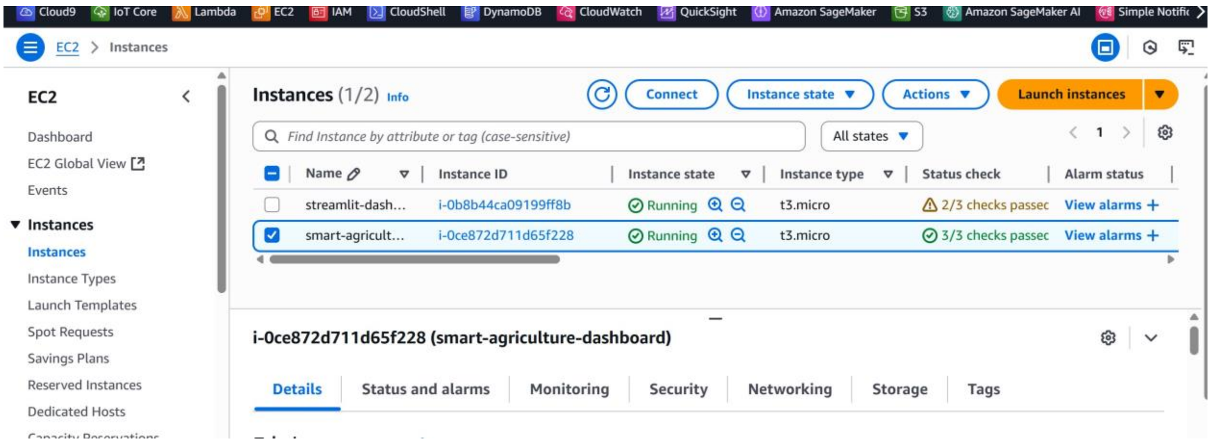
5 Server (EC2) Provisioning & System Setup

Amazon EC2 Installation

Sign up an EC2 instance (Ubuntu 22.04) to run the Streamlit dashboard. Opened a security group port 8501 to open up dashboard.

Set up Python packages and released the Streamlit application to monitor in realtime IoT and prediction.

Sign up an EC2 instance (Ubuntu 22.04) to run the Streamlit dashboard.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status
streamlit-dash...	i-0b8b44ca09199ff8b	Running	t3.micro	2/3 checks passec	View alarms +
smart-agricult...	i-0ce872d711d65f228	Running	t3.micro	3/3 checks passec	View alarms +

i-0ce872d711d65f228 (smart-agriculture-dashboard)

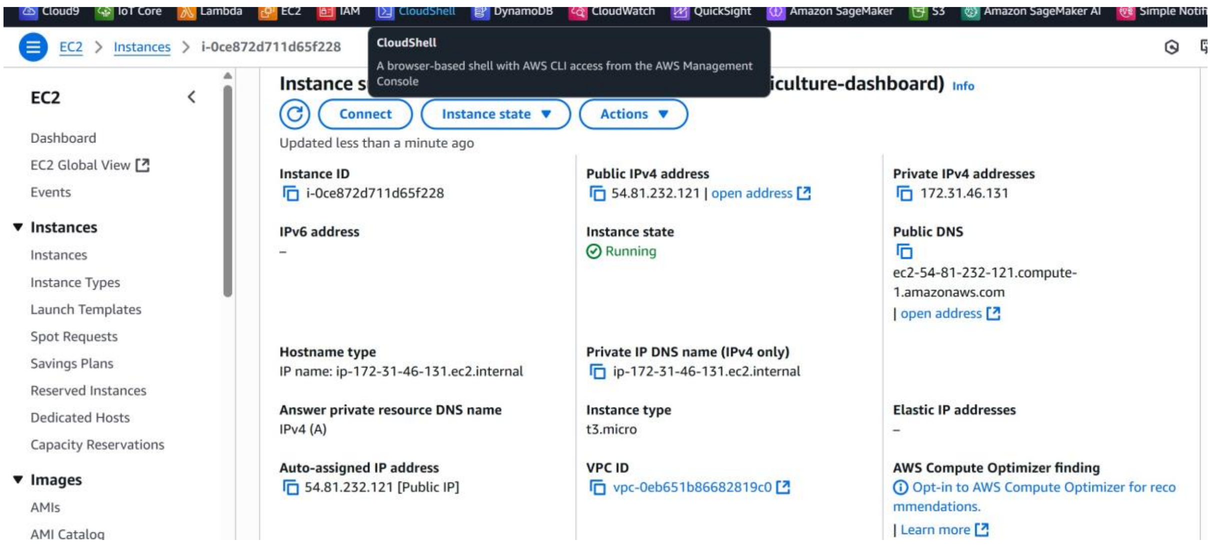
Figure 3: AWS EC2

Opened a security group port 8501 to open up dashboard.

Set up Python packages and released the Streamlit application to monitor in realtime IoT and prediction.

5.1 setup for dashboard in EC2

```
~ $ ssh –i ~/mynew.pem ec2–user@54 .81.232.121
```


Cloud9 · IoT Core · Lambda · EC2 · IAM · CloudShell · DynamoDB · CloudWatch · QuickSight · Amazon SageMaker · S3 · Amazon SageMaker AI · Simple Notif
EC2 > Instances > i-0ce872d711d65f228
CloudShell
A browser-based shell with AWS CLI access from the AWS Management Console
EC2
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Connect
Instance state
Actions

Instance s ...iculture-dashboard) Info

Updated less than a minute ago

Instance ID i-0ce872d711d65f228	Public IPv4 address 54.81.232.121 | open address	Private IPv4 addresses 172.31.46.131
IPv6 address –	Instance state Running	Public DNS ec2-54-81-232-121.compute-1.amazonaws.com | open address
Hostname type IP name: ip-172-31-46-131.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-46-131.ec2.internal	
Answer private resource DNS name IPv4 (A)	Instance type t3.micro	Elastic IP addresses –
Auto-assigned IP address 54.81.232.121 [Public IP]	VPC ID vpc-0eb651b86682819c0	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more

Figure 4: ec2 code

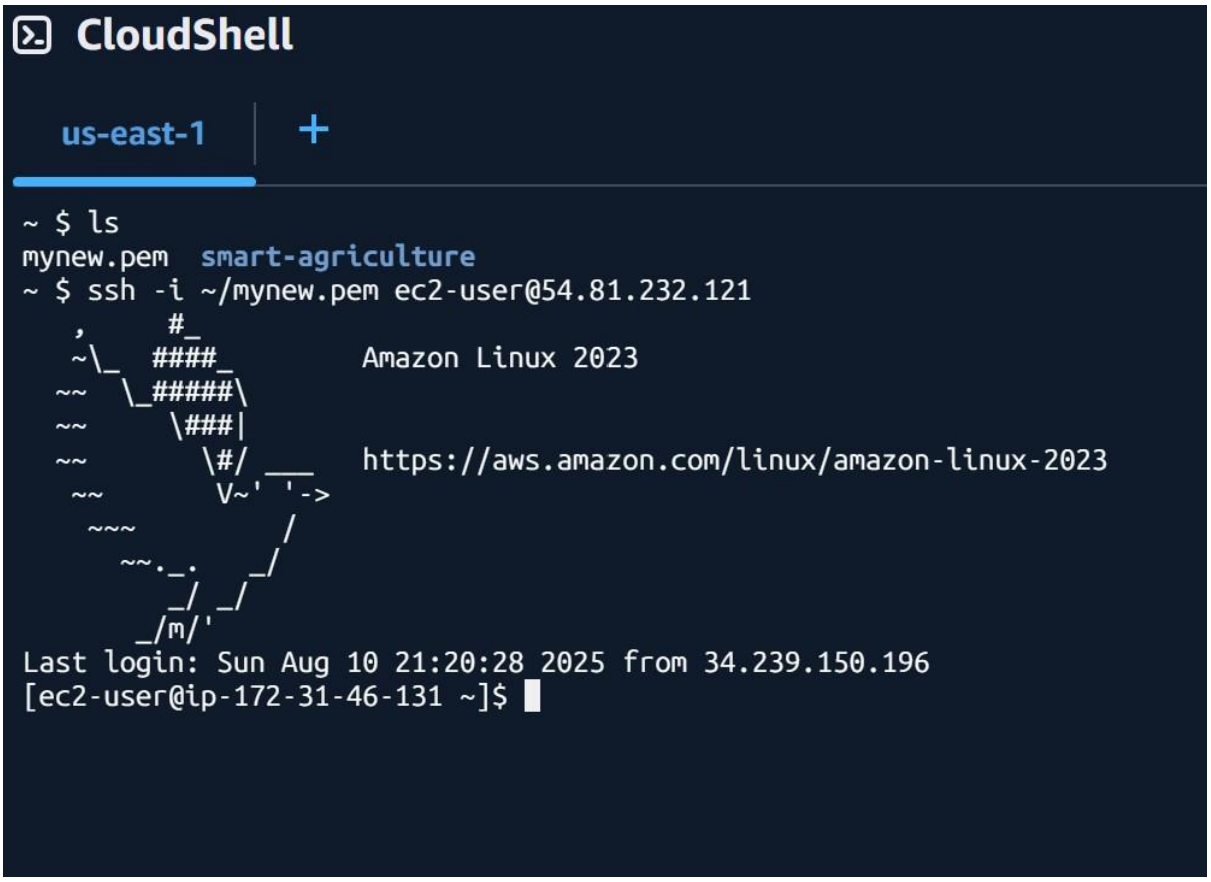
5.2 Base packages

```
[ ec2–user@ip –172–31–46–131 ~]$ ls
AmazonRootCA1.pem cert .pem dash . log dashboard . py device1_simulation nohup. out private . ke
public . key simulation . log
```

5.3 For dashboard

```
[ ec2−user@ip −172−31−46−131 ~]$ nohup streamlit run ~/dashboard . py −−ser [1] 51576
    [ ec2−user@ip −172−31−46−131 ~]$ nohup:       ignoring         input and appending outp

[1]+     Exit 1                                 nohup streamlit run ~/dashboard . py −−serv
```



```
~ $ ls
mynew.pem  smart-agriculture
~ $ ssh -i ~/mynew.pem ec2-user@54.81.232.121
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sun Aug 10 21:20:28 2025 from 34.239.150.196
[ec2-user@ip-172-31-46-131 ~]$ 
```

Figure 5: AWS Management Console

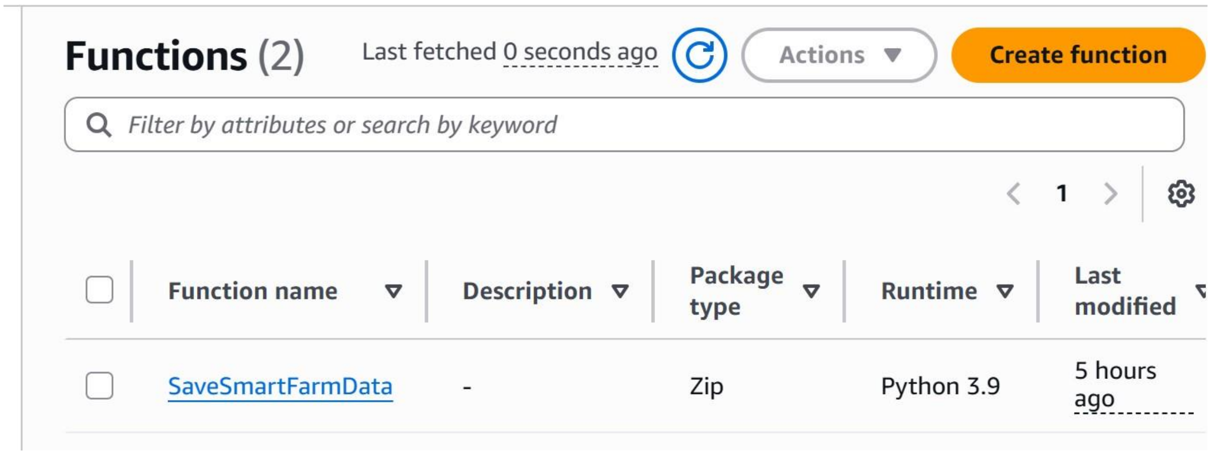
6 Lambda and AWS IOT triggered

Replace with actual DB credentials if using Postgres/MySQL.

7 AWS SageMAker Setup

Create new notebook after creating open jupiter :

```
import   pandas   as   pd
import numpy as np import
joblib import t a r f i l e
```



Functions (2) Last fetched 0 seconds ago Actions Create function

Filter by attributes or search by keyword

Function name	Description	Package type	Runtime	Last modified
SaveSmartFarmData	-	Zip	Python 3.9	5 hours ago

Figure 6: AWS Management Console

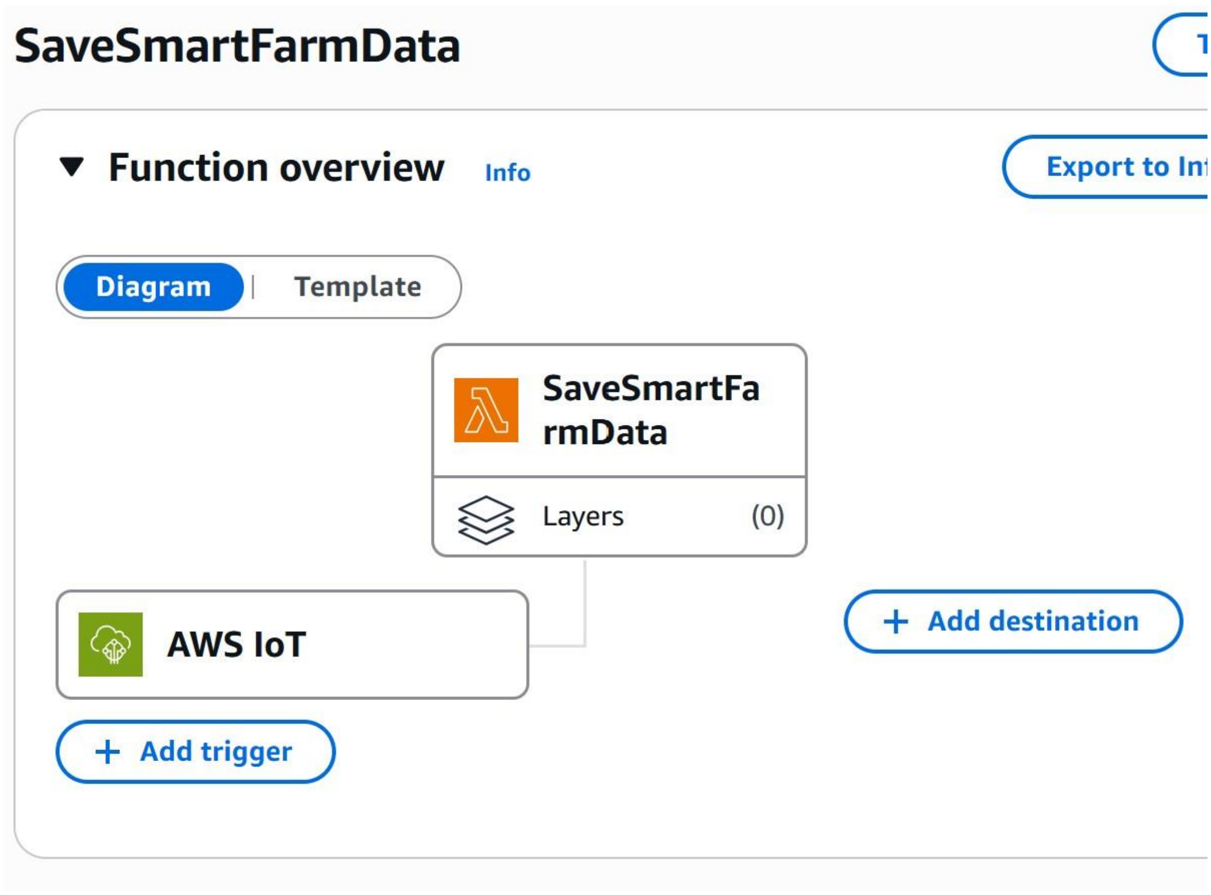
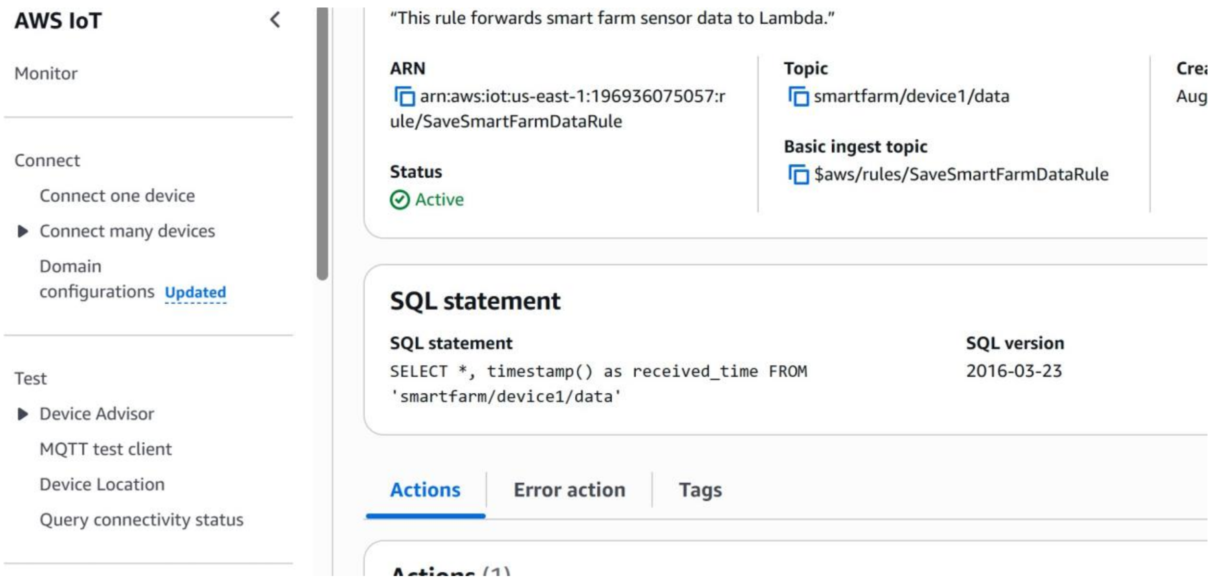


Figure 7: lambda

import sagemaker
from sklearn . model_selection import train_test_split from sklearn . ensemble import RandomForestClassifier from sklearn . metrics import accuracy_score , classification_report from sagemaker . sklearn . model import SKLearnModel from sagemaker import get_execution_role import boto3



"This rule forwards smart farm sensor data to Lambda."

ARN
arn:aws:iot:us-east-1:196936075057:rule/SaveSmartFarmDataRule

Topic
smartfarm/device1/data

Basic ingest topic
$aws/rules/SaveSmartFarmDataRule

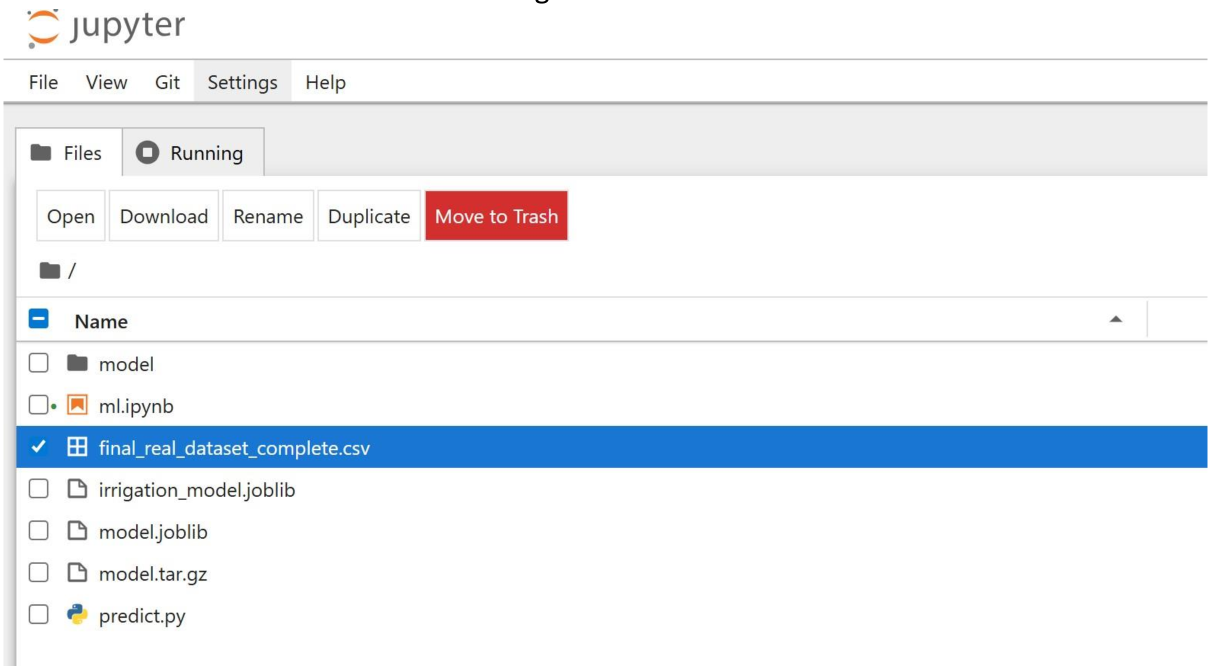
Status
Active

SQL statement

SQL statement
```
SELECT *, timestamp() as received_time FROM
'smartfarm/device1/data'
```

SQL version
2016-03-23

Figure 8: aws



model
ml.ipynb
final_real_dataset_complete.csv
irrigation_model.joblib
model.joblib
model.tar.gz
predict.py

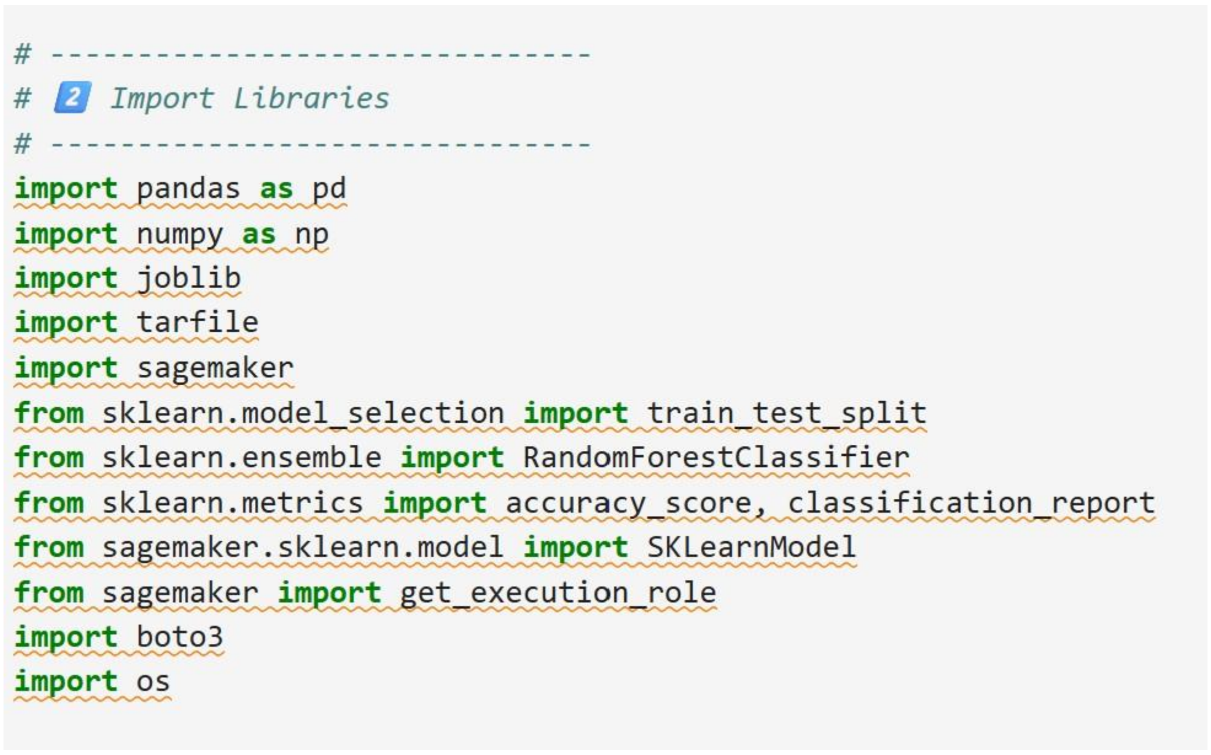
Figure 9: jupitur notebook

import os

7.1 load dataset

CSV_PATH = "final_real_dataset_complete . csv" *# must be in working dire* df = pd. read_csv(CSV_PATH

print("Dataset␣shape : " , df . shape) print("Columns : " , df .
columns . t o l i s t ())
display (df . head ())



```
# ------------------------------
# 2 Import Libraries
# ------------------------------
import pandas as pd
import numpy as np
import joblib
import tarfile
import sagemaker
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sagemaker.sklearn.model import SKLearnModel
from sagemaker import get_execution_role
import boto3
import os
```

Figure 10: libraries

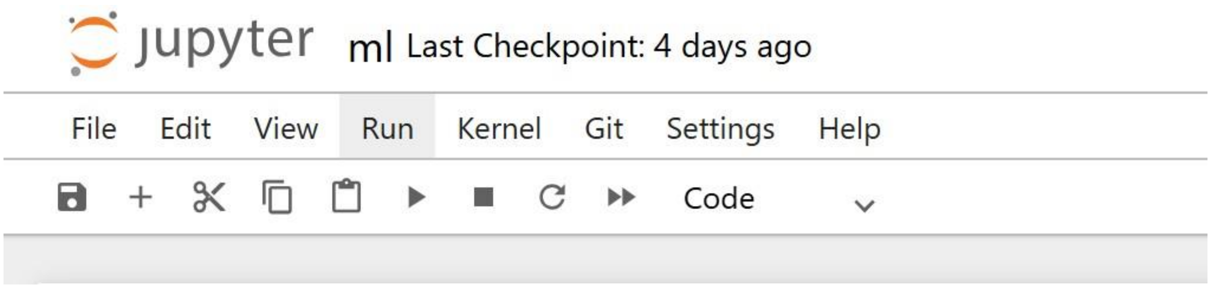


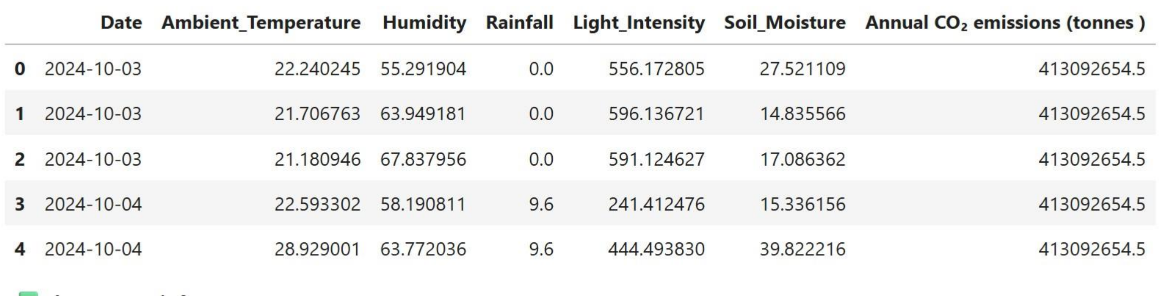
Figure 11: load

7.2 Train the Dataset

8 Deploy

model = RandomForestClassifier (
    n_estimators=100, random_state=42,
    class_weight=’ balanced ’
)
  model . f i t (X_train ,   y_train)

# Evaluate y_pred = model . predict (X_test)
print("Accuracy : " , accuracy_score (y_test , y_pred)) print(" Classification ␣Report :\n" , classification_report (y_test , y_pred) print("Predicted␣counts :\n" , pd. Series (y_pred). value_counts ())



	Date	Ambient_Temperature	Humidity	Rainfall	Light_Intensity	Soil_Moisture	Annual CO_2 emissions (tonnes)
0	2024-10-03	22.240245	55.291904	0.0	556.172805	27.521109	413092654.5
1	2024-10-03	21.706763	63.949181	0.0	596.136721	14.835566	413092654.5
2	2024-10-03	21.180946	67.837956	0.0	591.124627	17.086362	413092654.5
3	2024-10-04	22.593302	58.190811	9.6	241.412476	15.336156	413092654.5
4	2024-10-04	28.929001	63.772036	9.6	444.493830	39.822216	413092654.5

Figure 12: dataset

8.1 Deploy

```
sk_model = SKLearnModel( model_data=model_data_s3 ,
        role=ROLE,
           entry_point=ENTRY_POINT_NAME,            # our   robust   I/O script
        framework_version=FRAMEWORK_VERSION, py_version="py3" ,
        sagemaker_session=sm_session ,
)

predictor = sk_model . deploy ( initial_instance_count=1,
        instance_type=INSTANCE_TYPE
)
endpoint_name = predictor . endpoint_name print("␣Deployed␣endpoint : " ,
endpoint_name)
```

9 Final Deployment dashboard

In the cloudshell inside the EC2 bash

```
import os import ast import re import
time from decimal import Decimal

import pandas as pd import
streamlit as st

# ========== Settings ==========
LOG_PATH = "/home/ec2−user/simulation . log"
REFRESH_RATE = 5 # seconds between auto−refreshes PAGE_TITLE =
"␣Smart␣Farm␣Dashboard" PAGE_ICON = " "

st . set_page_config ( page_title=PAGE_TITLE, page_icon=PAGE_ICON, layout=" dashboard
```

urlhttp://http://54.81.232.121:8502/ in a browser.

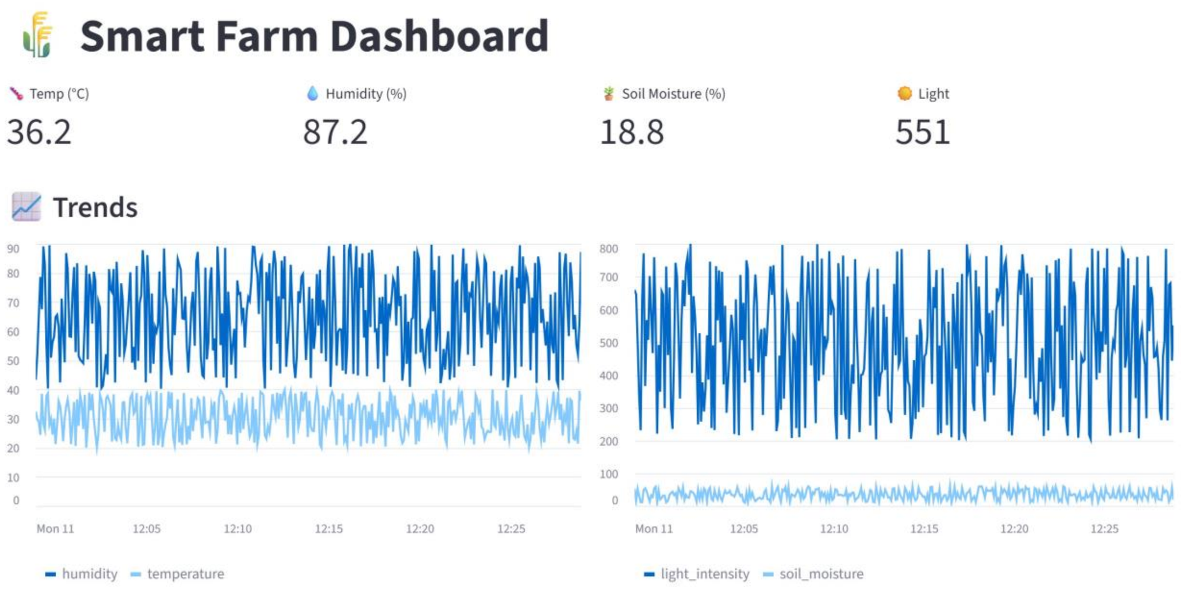


Figure 13: dashboard

```
 [ ec2−user@ip −172−31−46−131 ~]$ nohup streamlit run ~/dashboard . py −−server . [1] 51576
[ ec2−user@ip −172−31−46−131 ~]$ nohup:              ignoring             input and appending output t

[1]+        Exit 1                                          nohup streamlit run ~/dashboard . py −−server . p
 [ ec2−user@ip −172−31−46−131 ~]$
```

9.1 Pushing Code to GitHub

```
# Check current branch git branch

# Stage all changes git add .

# Commit with a message git commit −m "Added
 dashboard . py"

#  Push  to  GitHub  git  push
 origin main
```

10 References

- linux: https://docs.kernel.org/
- GitHub Actions: https://docs.github.com/
- AWS : https://us-east-1.console.aws.amazon.com/
- Random forest algorithm https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

- python libraries https://docs.python.org/3/

graphicx tikz [absolute,overlay]textpos