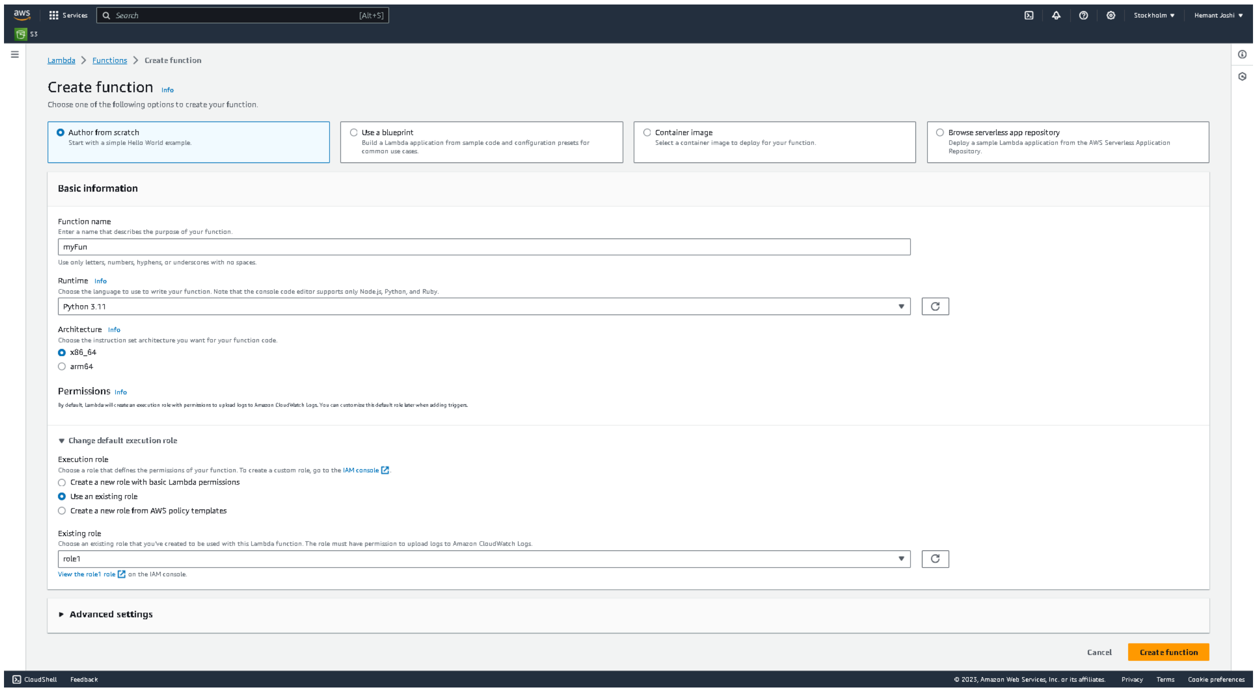
btp documentation

step1: setting up lambda service::

go to lambda service of amazon aws and there createa new function



Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

- [✓ Author from scratch] Start with a simple Hello World example.
- Use a blueprint: Build a Lambda application from sample code and configuration presets for common use cases.
- Container image: Select a container image to deploy for your function.
- Browse serverless app repository: Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

Function name
Enter a name that describes the purpose of your function.
myFun
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.11

Architecture Info
Choose the instruction set architecture you want for your function code.
[✓ x86_64] arm64

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
- Create a new role with basic Lambda permissions
- [✓ Use an existing role]
- Create a new role from AWS policy templates

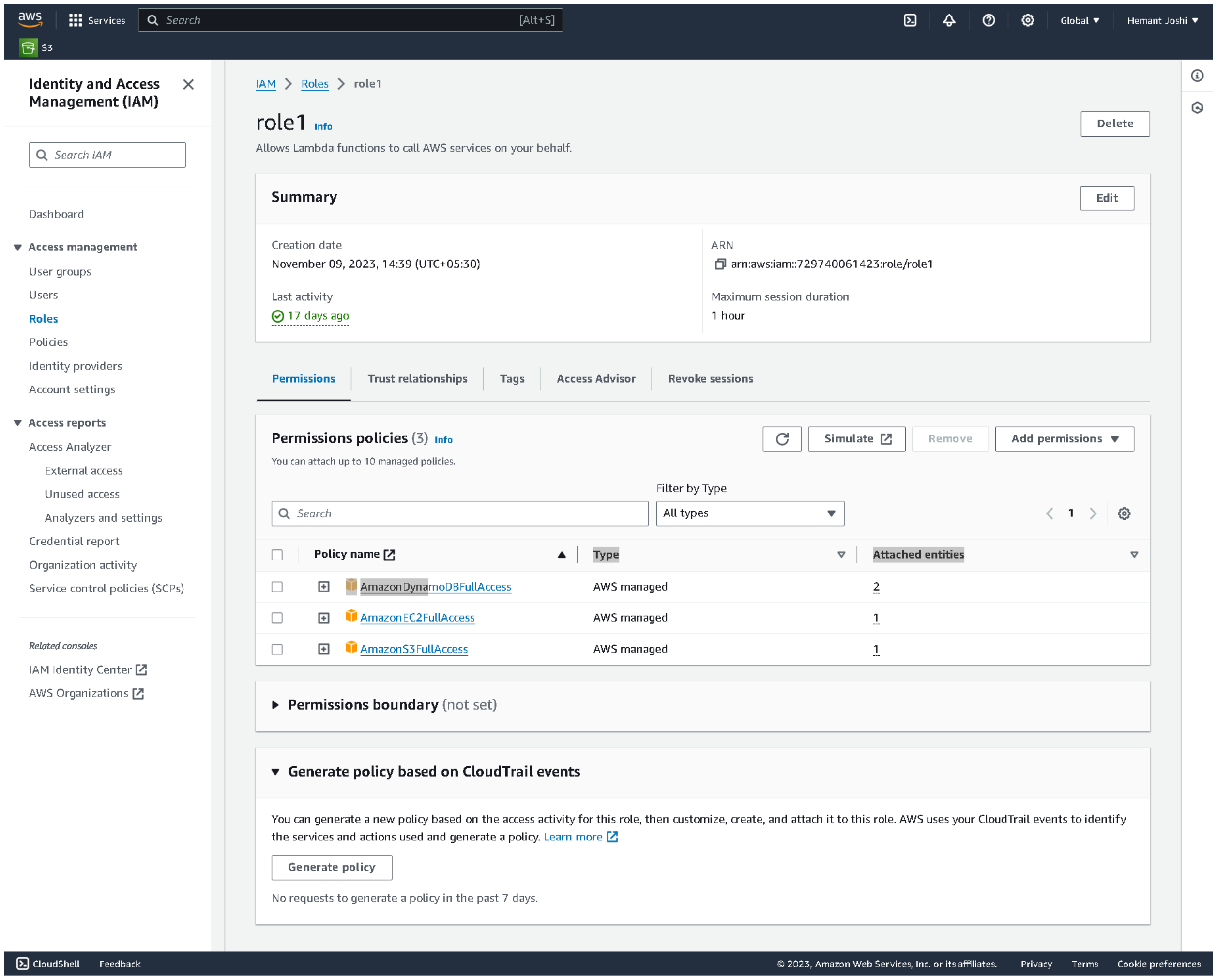
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
role1
View the role1 role on the IAM console.

▶ Advanced settings

Cancel Create function

Under execution role choose the role (permissions) which you want to give to your lambda code

to create new role click create a new role and give following permission



IAM > Roles > role1

role1 Info

Allows Lambda functions to call AWS services on your behalf.

Summary

Creation date: November 09, 2023, 14:39 (UTC+05:30)

ARN: arn:aws:iam::729740061423:role/role1

Last activity: 17 days ago

Maximum session duration: 1 hour

Permissions policies (3)

You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	2
AmazonEC2FullAccess	AWS managed	1
AmazonS3FullAccess	AWS managed	1

Permissions boundary (not set)

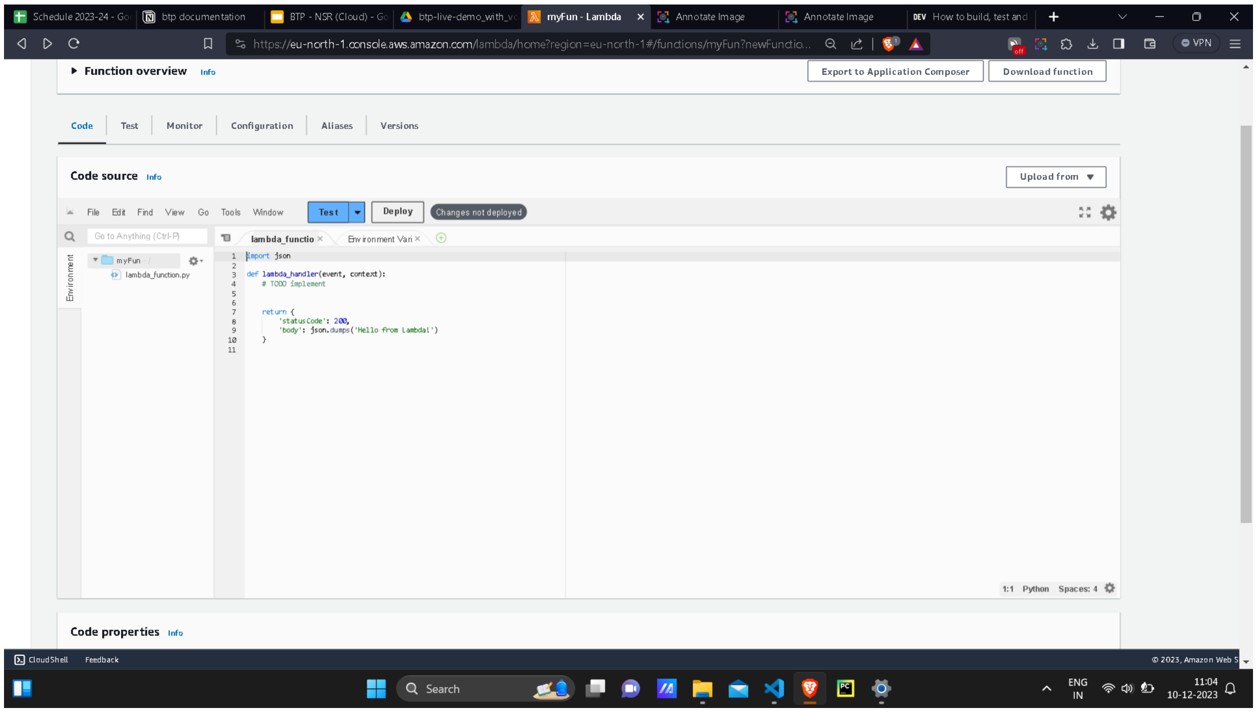
Generate policy based on CloudTrail events

You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. Learn more

No requests to generate a policy in the past 7 days.

now click on create a new function in the bottom right corner

paste your lambda code here (lambda code transfers the dynamodb data to s3 bucket)

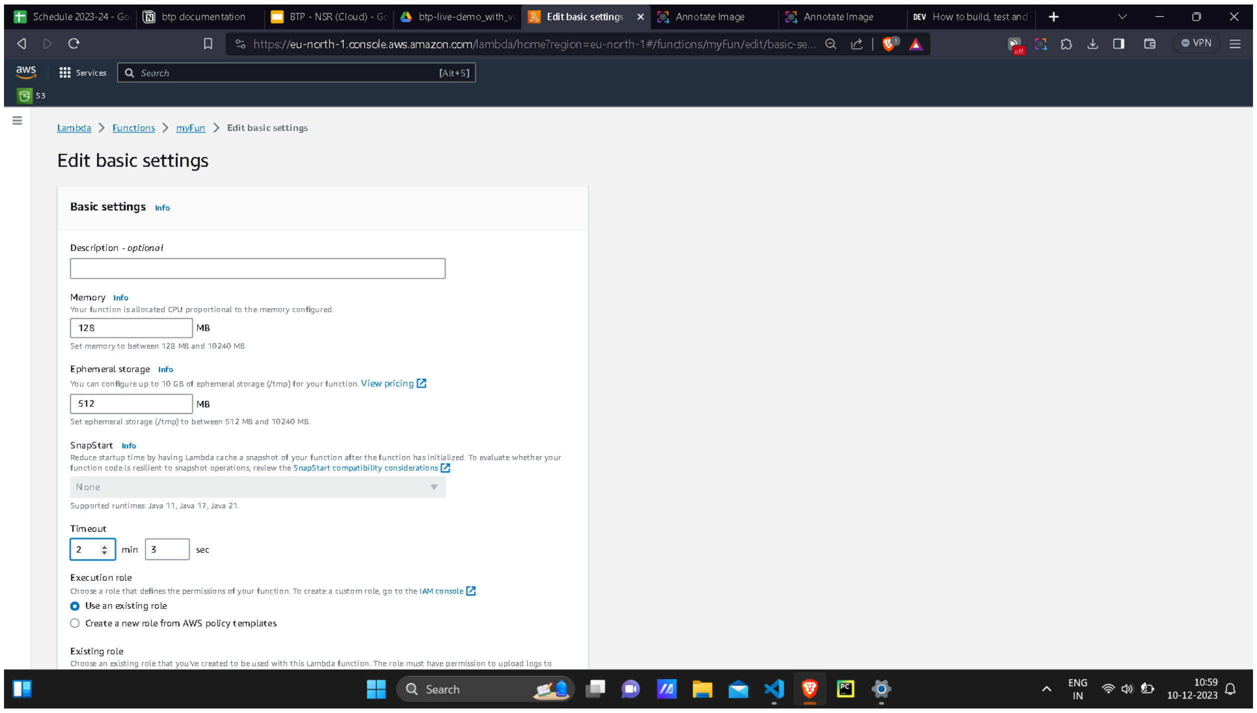


```python
import json

def lambda_handler(event, context):
    # TODO implement


    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }

```

lambda code ::

```
import json
import boto3
import csv
import io
def lambda_handler(event, context):
    # Create an S3 client
    s3_client = boto3.client('s3')

    # Create a DynamoDB client
    dynamodb_client = boto3.client('dynamodb')

    # Specify the name of your DynamoDB table
    table_name = 'mq_sensor_data_final'

    # Specify the name of your S3 bucket
    bucket_name = 'btp9785'

    try:
        # Retrieve the entire table from DynamoDB
        response = dynamodb_client.scan(TableName=table_name)

        # Convert the response to a list of dictionaries
        items = response['Items']

        # Define the field names for the CSV file
        field_names = items[0].keys()
        # Create a CSV file in memory
        csv_file = io.StringIO()
        writer = csv.DictWriter(csv_file, fieldnames=field_names)

        # Write the field names to the CSV file
        writer.writeheader()

        # Write the data to the CSV file
        writer.writerows(items)

        # Save the CSV file to an S3 object
        s3_client.put_object(Body=csv_file.getvalue(), Bucket=bucket_name, Key='dynamodb_data.csv')

        return {
            'statusCode': 200,
            'body': json.dumps('Data saved to S3 successfully in CSV format!')
        }
    except Exception as e:
        return {
            'statusCode': 500,
            'body': json.dumps('Error: {}'.format(str(e)))
        }
```

Under configuration tab set the execution time limit to 2 minutes


Lambda > Functions > myFun > Edit basic settings

Edit basic settings

Basic settings Info

Description - *optional*

Memory Info
Your function is allocated CPU proportional to the memory configured.
128 MB
Set memory to between 128 MB and 10240 MB

Ephemeral storage Info
You can configure up to 10 GB of ephemeral storage (/tmp) for your function. View pricing
512 MB
Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart Info
Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the SnapStart compatibility considerations.
None
Supported runtimes: Java 11, Java 17, Java 21.

Timeout
2 min 3 sec

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
[✓ Use an existing role]
Create a new role from AWS policy templates

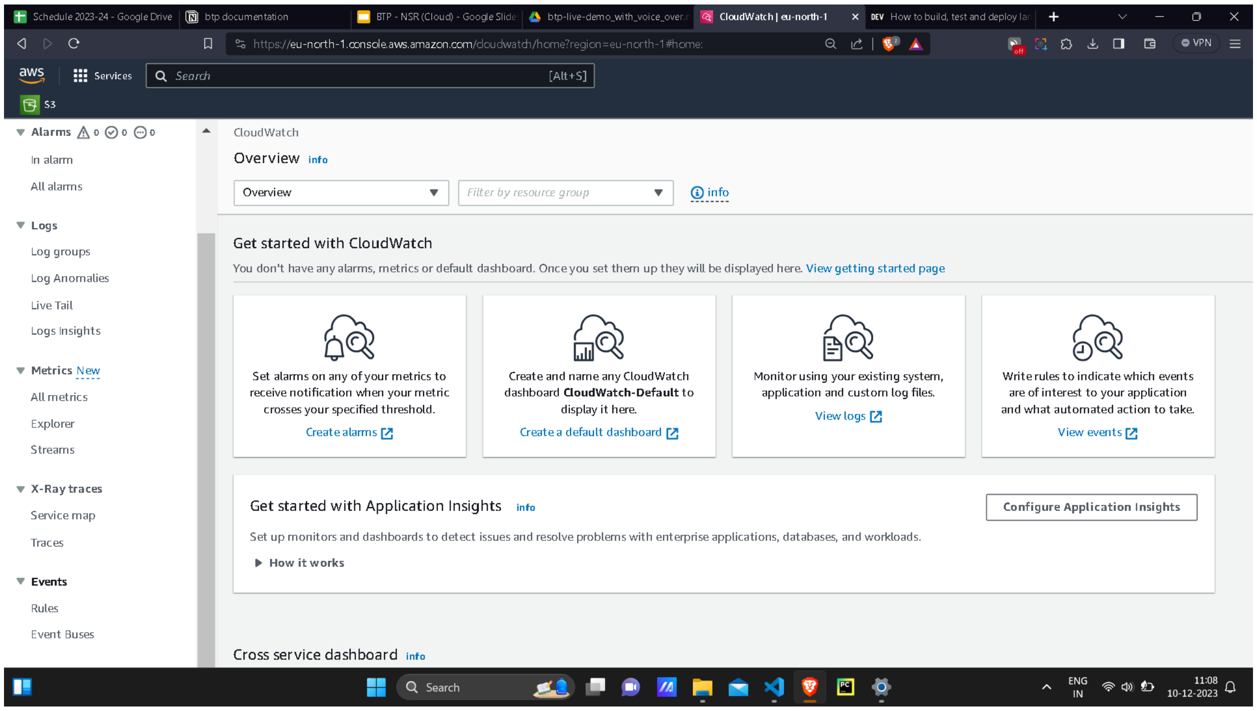
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to

now test and deploy your lambda code

Scheduling lambda service at 1 minute of interval

go to aws→cloudwatch

there under events (bottom left) go to rules



Get started with CloudWatch

You don't have any alarms, metrics or default dashboard. Once you set them up they will be displayed here. View getting started page

Set alarms on any of your metrics to receive notification when your metric crosses your specified threshold. Create alarms

Create and name any CloudWatch dashboard **CloudWatch-Default** to display it here. Create a default dashboard

Monitor using your existing system, application and custom log files. View logs

Write rules to indicate which events are of interest to your application and what automated action to take. View events

Get started with Application Insights info

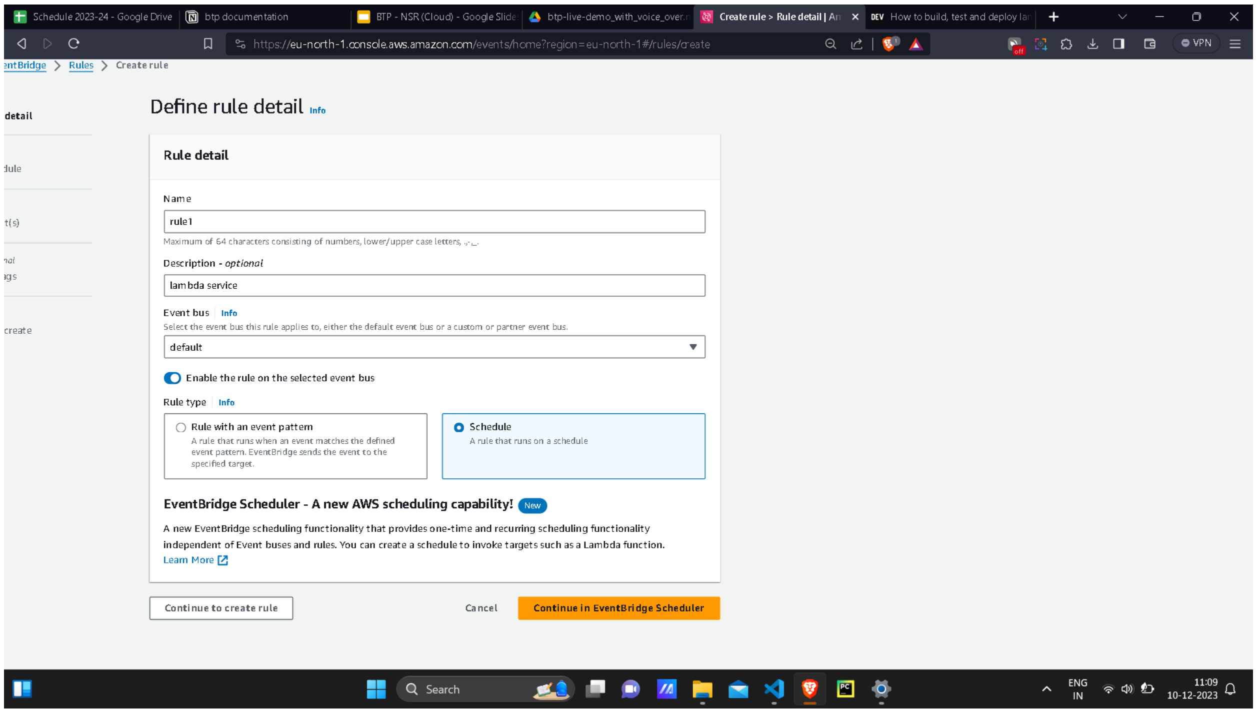
Configure Application Insights

Set up monitors and dashboards to detect issues and resolve problems with enterprise applications, databases, and workloads.

How it works

Cross service dashboard info

setup the rule details and then click on Continue to create rule


https://eu-north-1.console.aws.amazon.com/events/home?region=eu-north-1#/rules/create
EventBridge > Rules > Create rule

Define rule detail Info

Rule detail

Name

rule1

Maximum of 64 characters consisting of numbers, lower/upper case letters, .,-,_.

Description - *optional*

lambda service

Event bus Info

Select the event bus this rule applies to, either the default event bus or a custom or partner event bus.

default

Enable the rule on the selected event bus

Rule type Info

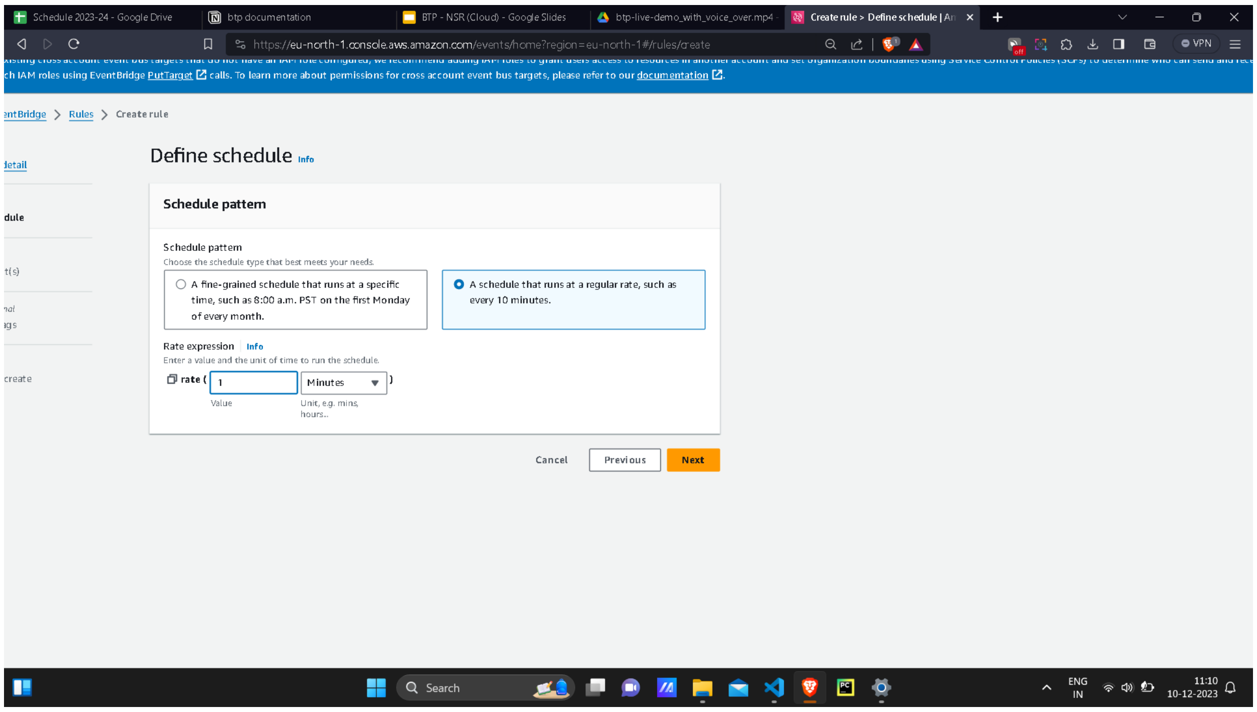
- Rule with an event pattern — A rule that runs when an event matches the defined event pattern. EventBridge sends the event to the specified target.
- [✓ Schedule] — A rule that runs on a schedule

EventBridge Scheduler - A new AWS scheduling capability! New

A new EventBridge scheduling functionality that provides one-time and recurring scheduling functionality independent of Event buses and rules. You can create a schedule to invoke targets such as a Lambda function.

Learn More

set the timeperiod of lambda function to 1 minute


https://eu-north-1.console.aws.amazon.com/events/home?region=eu-north-1#/rules/create
EventBridge > Rules > Create rule

ch IAM roles using EventBridge PutTarget calls. To learn more about permissions for cross account event bus targets, please refer to our documentation.

Define schedule Info

Schedule pattern

Schedule pattern

Choose the schedule type that best meets your needs.

- A fine-grained schedule that runs at a specific time, such as 8:00 a.m. PST on the first Monday of every month.
- [✓ A schedule that runs at a regular rate, such as every 10 minutes.]

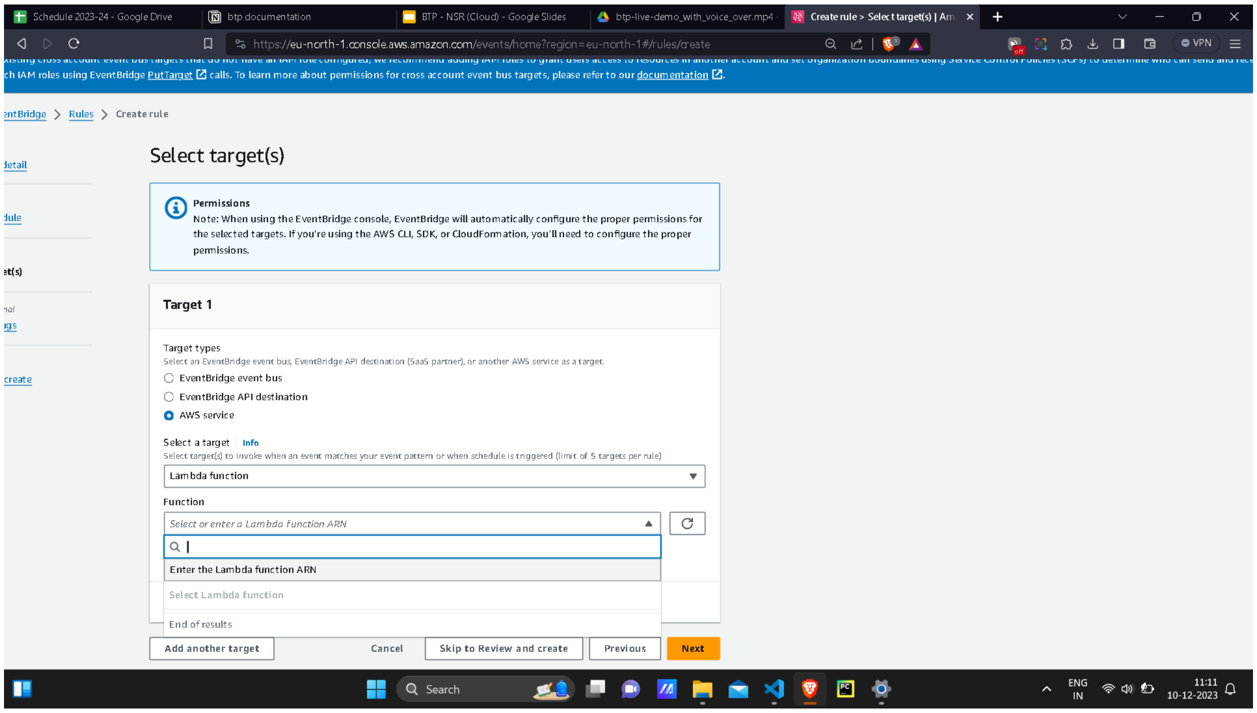
Rate expression Info

Enter a value and the unit of time to run the schedule.

rate (1 Minutes)

Value — Unit, e.g. mins, hours...

select the service which you want to use for this rule (lambda service here)


Select target(s)

Permissions
Note: When using the EventBridge console, EventBridge will automatically configure the proper permissions for the selected targets. If you're using the AWS CLI, SDK, or CloudFormation, you'll need to configure the proper permissions.

Target 1

Target types
Select an EventBridge event bus, EventBridge API destination (SaaS partner), or another AWS service as a target.
- EventBridge event bus
- EventBridge API destination
- [✓ AWS service]

Select a target
Select target(s) to invoke when an event matches your event pattern or when schedule is triggered (limit of 5 targets per rule)
Lambda function

Function
Select or enter a Lambda function ARN
Enter the Lambda function ARN
Select Lambda function
End of results

https://eu-north-1.console.aws.amazon.com/events/home?region=eu-north-1#/rules/create
EventBridge > Rules > Create rule
Add another target
Cancel
Skip to Review and create
Previous
Next

click on next and your rule is created

setting up of ec2 instance :

website code

```
import streamlit as st
import pandas as pd
import time
import joblib
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from PIL import Image




import streamlit as st

# Set the heading
st.markdown("<h1>Btp Project supervised by N S Rajput</h1>", unsafe_allow_html=True)
```

```
model = joblib.load('model.joblib')




prediction_box = st.empty()
class_box = st.empty()
arr_box  =  st.empty()
shape_box = st.empty()





import ast

def reorder_columns(df):
    try:
      a.drop('Time',axis='columns',inplace=True)

      a.drop('Unnamed: 0',axis='columns',inplace=True)

    except Exception as e:
      print('...')
    new_column_order = ['MQ2', 'MQ3', 'MQ4', 'MQ6', 'MQ7', 'MQ8', 'MQ9', 'MQ135']
    assert set(new_column_order) == set(df.columns), "Invalid column order"
    reordered_df = df[new_column_order]

    return reordered_df



def extract1(dict_string):
    try:
        # Safely evaluate the string as a Python literal (dictionary)
        dictionary = ast.literal_eval(dict_string)

        # Extract the value associated with the key 'N'
        value = dictionary.get('N', None)

        # Convert the value to an integer (if it's a valid integer)
        if value is not None:
            try:
                numeric_value = int(value)
                return numeric_value
            except ValueError:
                return None
        else:
            return None
    except (ValueError, SyntaxError):
        # Handle errors in case the string is not a valid dictionary
        return None

st.markdown(
    """
    <style>
    body {
        background-color: #222;
        color: white;
    }
    </style>
    """,
    unsafe_allow_html=True
)
#

# Set the members of the team
members = ["Hemant Joshi", "Vedanth  Powar", "Prasasthi Tripathi", "Akshit Singhla"]

# Display the members in the bottom rightmost corner
st.sidebar.title("Team Members")
for member in members:
    st.sidebar.write(member)




# Function to read the CSV file from the S3 bucket URL
def read_csv_from_s3(url):
    df = pd.read_csv(url)
    return df

# Set the URL of the CSV file in the S3 bucket
csv_url = "https://btp9785.s3.eu-north-1.amazonaws.com/dynamodb_data.csv"





lr = 0

while True:
    # Wait for 30 seconds
    time.sleep(5)

    # Read the CSV file from the S3 bucket URL
    try:
        # Attempt to read the CSV file from the provided URL
        # a = pd.read_csv(csv_url)
        dff = pd.read_csv(csv_url)
        # st.write(dff)
        dff = dff.sort_values(by='Time')
        a = dff
        # st.write(dff)

        for column in a.columns:
            a[column] = a[column].apply(extract1)
        a = reorder_columns(a)
        # st.write(a)
        # Clear the previous data and display the updated data




        arr = np.array(a.iloc[-1])

        # st.write(arr)
        predi = model.predict([arr])
```

```
#
        strr = 'air'
        if predi==0:
            strr = "Incense Stick 🔥🔥🔥"
        if predi==1:
            strr = 'Camphor'
        if predi==2:
            strr = 'Air'


        print(arr)
        class_box.markdown(f"""
   <div style="border: 1px solid #000; padding: 10px; width: 100%; height: 100%; margin: 0 auto;">
       <font size="800"> {strr}</font>
   </div>
   """, unsafe_allow_html=True)
        # img_box.image(img)



        # prediction_box.markdown(f'Model prediction: {model.predict([arr])}')
        rr = a.shape[0]
        if lr!=rr:
            # st.write('a')
            # st.write(a)

            # shape_box.write(a.shape)
            # arr_box.write(arr)
            lr = a.shape[0]
            # st.write('dff')
            # st.write(dff)

    except Exception as e:
    # Handle any exceptions that occur during the execution
        st.error(f"An error occurred: {str(e)}")
```

upload your website code and model to ec2 instance.

to transfer files between your computer and EC2 server you can use WinSCP.

run your website using streamlit run website_code.py

model file

```
# -*- coding: utf-8 -*-
"""btp final last.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1Bf0EKm7nJEFC8oEppd7tJduNqI1JIPC9
"""
```

```
import numpy as np
import pandas as pd
import keras
from sklearn import preprocessing
import tensorflow as tf
from sklearn import preprocessing
import matplotlib
import matplotlib.pyplot as plt
# import seaborn as sns % matplotlib inline

agarbati_data = pd.read_csv('/content/AGARBATTI3.csv')
camphor_data = pd.read_csv('/content/CAMPHOR.csv')
air1_data = pd.read_csv('/content/OPEN AFTER.csv')
air2_data = pd.read_csv('/content/OPEN AIR.csv')

# agarbati_data.drop('Time',axis='columns',inplace=True)
# camphor_data.drop('Time',axis='columns',inplace=True)
# air1_data.drop('Time',axis='columns',inplace=True)
# air2_data.drop('Time',axis='columns',inplace=True)

agarbati_data['Gas'] = 0
camphor_data['Gas'] = 1
air1_data['Gas'] =  2
air2_data['Gas'] = 2

# scaler = preprocessing.MinMaxScaler()
# air_data = scaler.fit_transform(air_data)
# scaler = preprocessing.MinMaxScaler()
# ele_data = scaler.fit_transform(ele_data)
# scaler = preprocessing.MinMaxScaler()
# oil_data = scaler.fit_transform(oil_data)
# scaler = preprocessing.MinMaxScaler()
# solid_data = scaler.fit_transform(solid_data)

df = pd.concat([agarbati_data, camphor_data, air1_data, air2_data], ignore_index=True)
# df = np.array(df)
from sklearn.utils import shuffle
shuffle(df)

# df = np.array(df)

X = df.drop(['Gas'], axis=1)

y = df['Gas']
y

from sklearn import preprocessing
lc = preprocessing.LabelEncoder()
y = lc.fit_transform(df['Gas'])

y = pd.DataFrame(y,columns=['Gas'])
X  = X
np.max(y)

# split data into training and testing sets
X = np.array(X)
y = np.array(y)
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

```
from sklearn.ensemble import RandomForestClassifier



# instantiate the classifier

rfc = RandomForestClassifier(random_state=0)



# fit the model

rfc.fit(X_train, y_train)



# Predict the Test set results

y_pred = rfc.predict(X_test)



# Check accuracy score

from sklearn.metrics import accuracy_score

print('Model accuracy score with 10 decision-trees : {0:0.4f}'. format(accuracy_score(y_test, y_pred)))

# instantiate the classifier with n_estimators = 100

rfc_100 = RandomForestClassifier(n_estimators=100, random_state=0)



# fit the model to the training set

rfc_100.fit(X_train, y_train)



# Predict on the test set results

y_pred_100 = rfc_100.predict(X_test)



# Check accuracy score

print('Model accuracy score with 100 decision-trees : {0:0.4f}'. format(accuracy_score(y_test, y_pred_100)))

for i in range(20):
  print(rfc_100.predict([X_test[i]])," : ", y_test[i])

import joblib
y_pred = rfc_100.predict(X_test)
# # Assuming 'model' is your trained RandomForestClassifier
joblib.dump(rfc_100, 'model.joblib')
import matplotlib.pyplot as plt

import seaborn as sn
```

```
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_pred)

plt.figure(figsize = (10,4))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')

# loaded_model = joblib.load('model.joblib')

loaded_model.predict([X_train[0]])

df1 = pd.read_csv('/content/df.csv')

df1.head(5)
df1.iloc[0]

def extract_number(cell):
    try:
        return int(cell.get('N'))
    except (AttributeError, ValueError):
        return None

# Apply the function to each column in the DataFrame
for column in df1.columns:
    df1[column] = df1[column].apply(extract_number)

# Now, 'df1' contains the numerical parts extracted from the dictionaries
print(df1.head(5))

import ast

def extract1(dict_string):
    try:
        # Safely evaluate the string as a Python literal (dictionary)
        dictionary = ast.literal_eval(dict_string)

        # Extract the value associated with the key 'N'
        value = dictionary.get('N', None)

        # Convert the value to an integer (if it's a valid integer)
        if value is not None:
            try:
                numeric_value = int(value)
                return numeric_value
            except ValueError:
                return None
        else:
            return None
    except (ValueError, SyntaxError):
        # Handle errors in case the string is not a valid dictionary
        return None

dff= pd.read_csv('/content/df.csv')

a  =dff
for column in df1.columns:
    a[column] = a[column].apply(extract1)

a.head(5)

print(a.columns)
print(air_data.columns)

a.head(1)

def reorder_columns(df):
    try:

      a.drop('Unnamed: 0',axis='columns',inplace=True)
      a.drop('Time',axis='columns',inplace=True)
    except Exception as e:
      print('...')
    new_column_order = ['MQ135', 'MQ2', 'MQ3', 'MQ4', 'MQ6', 'MQ7', 'MQ8', 'MQ9']
    assert set(new_column_order) == set(df.columns), "Invalid column order"
    reordered_df = df[new_column_order]

    return reordered_df

b = reorder_columns(a)

a.head(1)

# import joblib
# import numpy as np
# import pandas as pd
# import keras
# from sklearn import preprocessing
# import tensorflow as tf
# from sklearn import preprocessing
# import matplotlib
# import matplotlib.pyplot as plt
# # import seaborn as sns % matplotlib inline
# arr = [429,530,526,435,518,430,522,429]
# arr = [437,535,531,411,534,441,533,441]
arr = [1024,592,590,1024,589,1024,558,400]
# # rfc_100 = joblib.load('/content/model.joblib')
print(rfc_100.predict([arr]))

joblib.dump(rfc_100,'model2.joblib')

rfc_100.predict([[533 ,534 ,598 ,533 ,598 ,534 ,600 ,598]])

cmper = pd.read_csv('/content/CAMPHOR.csv')
cmper
```

setting up of ec2 instance

go to aws→ec2→instances→create instance

imp things to do here

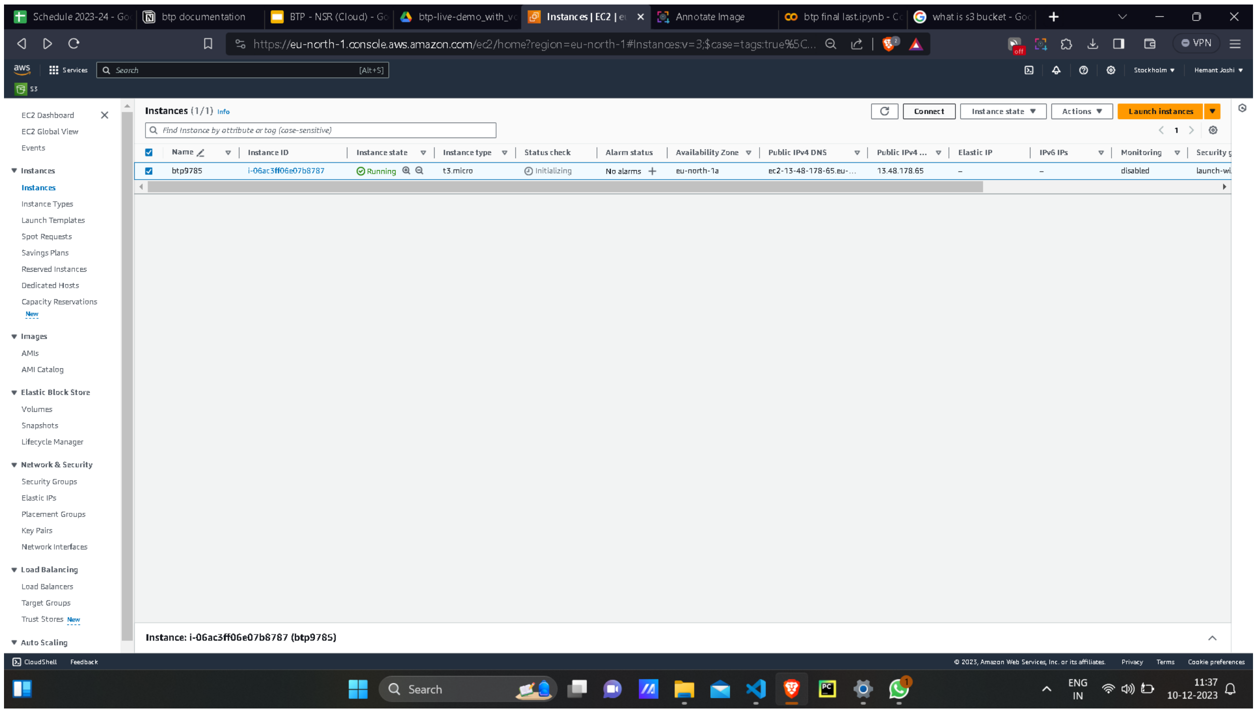
select ubuntu free tier container from list of containers

download key-pair file (this file will act as a credential when we try to connect our local pc to this server to transfer files) allow internet traffic to this server

select above settings and then launch the instance

then connect the server

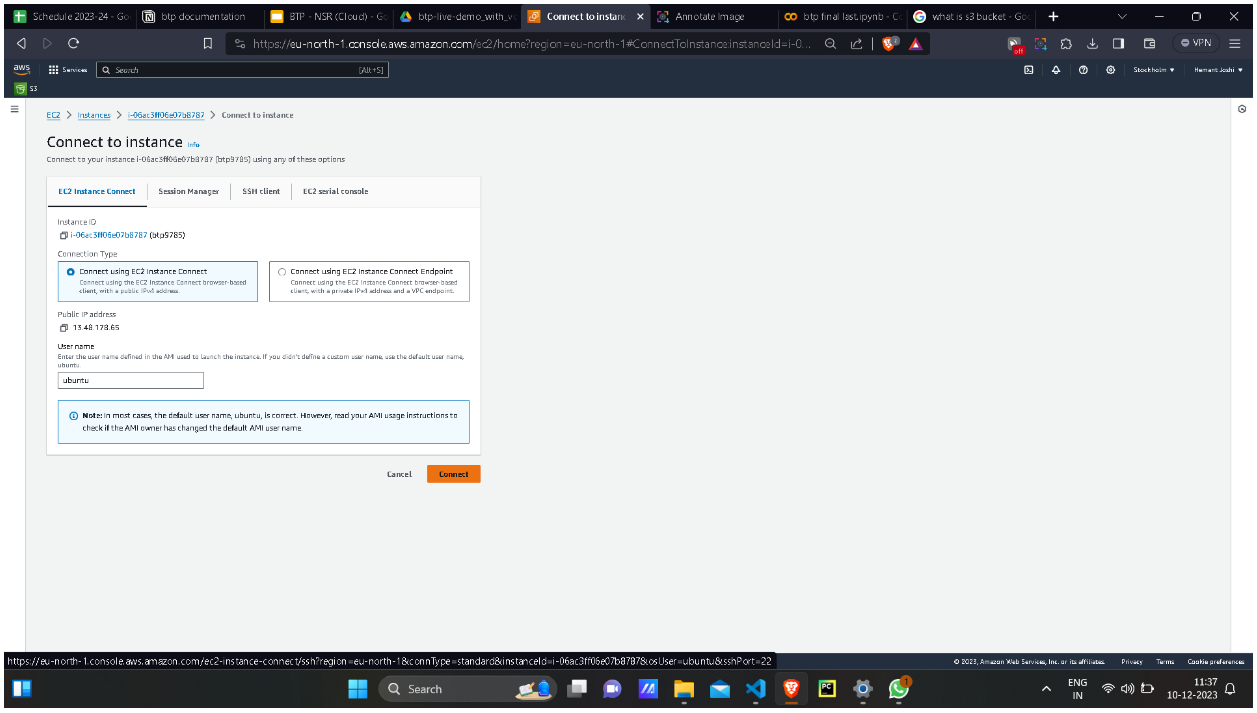
select the instance and click on connect on top right corner



Instances (1/1)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring	Security g
btp9785	i-06ac3ff06e07b8787	Running	t3.micro	Initializing	No alarms	eu-north-1a	ec2-13-48-178-65.eu-...	13.48.178.65	–	–	disabled	launch-wi

Instance: i-06ac3ff06e07b8787 (btp9785)


EC2 > Instances > i-06ac3ff06e07b8787 > Connect to instance

Connect to instance Info

Connect to your instance i-06ac3ff06e07b8787 (btp9785) using any of these options

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID
i-06ac3ff06e07b8787 (btp9785)

Connection Type
[✓ Connect using EC2 Instance Connect — Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.]
Connect using EC2 Instance Connect Endpoint — Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Public IP address
13.48.178.65

User name
Enter the user name defined in the AMI used to launch the instance. If you didn't define a custom user name, use the default user name, ubuntu.
ubuntu

Note: In most cases, the default user name, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel | Connect

https://eu-north-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=eu-north-1&connType=standard&instanceId=i-06ac3ff06e07b8787&osUser=ubuntu&sshPort=22

now transfer the website_code.py and model file to ec2 instance with help of winSCP server

use that key-pair file which we generated during creating of ec2 instance to connect to the server

install relevant libraries in ec2 and then you are good to go