英特尔® 边缘与 AWS Cloud 开展 AI 推理协作

我们介绍了边缘到云端架构的优势、由英特尔和 AWS 提供支持的示例模型以及更多帮助改善人类生活的用例。

边缘到云端架构的优势：要点

- 边缘安全性。数据隐私是许多行业（例如医疗保健和公共部门）的重大关切。边缘设备可以存储和加密敏感数据，并在需要时保护用户的隐私。
- 低延迟。在边缘启用 AI 推理的优势在于可避免往返云数据中心进行处理。您将获得近乎实时的分析和决策能力，不必担心网络上的数据拥塞、数据中心停电和其他事件。
- 更高效的数据工作负载分配。通过边缘层和云端层，开发人员可以决定在边缘或云端应处理多少数据工作负载。

立即开始

作者：

Vibhu Bithar
Chen Su
Devang Aggarwal

播放视频
播放

静音
已加载：0%

剩余时间-0:00
分享 画中画 全屏

介绍

2020 年是转型之年。全球新冠疫情从根本上改变了人们彼此互动的方式。在疫情加剧之际，保持社交距离对于我们创造安全的环境变得至关重要。通过在边缘部署 AI 和计算机视觉，英特尔和 AWS 团队推出了社交距离参考实现方案，开发人员只需通过一键安装和适当的定制操作便可在您当地的社区扩展该技术。在本博文中，我们介绍了边缘到云端架构的优势、由英特尔和 AWS 提供支持的示例模型以及英特尔® 边缘软件中心上可帮助改善人类生活的更多用例。

英特尔® 边缘与 AWS Cloud 协作

在新冠疫情爆发后，世界各地的许多医学专家均表示保持社交距离是预防这种疾病传播的最有效的非药物方法之一。

为了支持当前的疫情防控，英特尔推出了一种强大的参考实现方法，通过英特尔® OpenVINO™ 工具套件分发版实施计算机视觉推理，以测量人们之间的社交距离并将数据保存至 InfluxDB，进而帮助抑制疫情传播。这些数据可在 Grafana 仪表板上进行直观显示。

本博文介绍了该参考实现方案的安装、部署和定制信息。

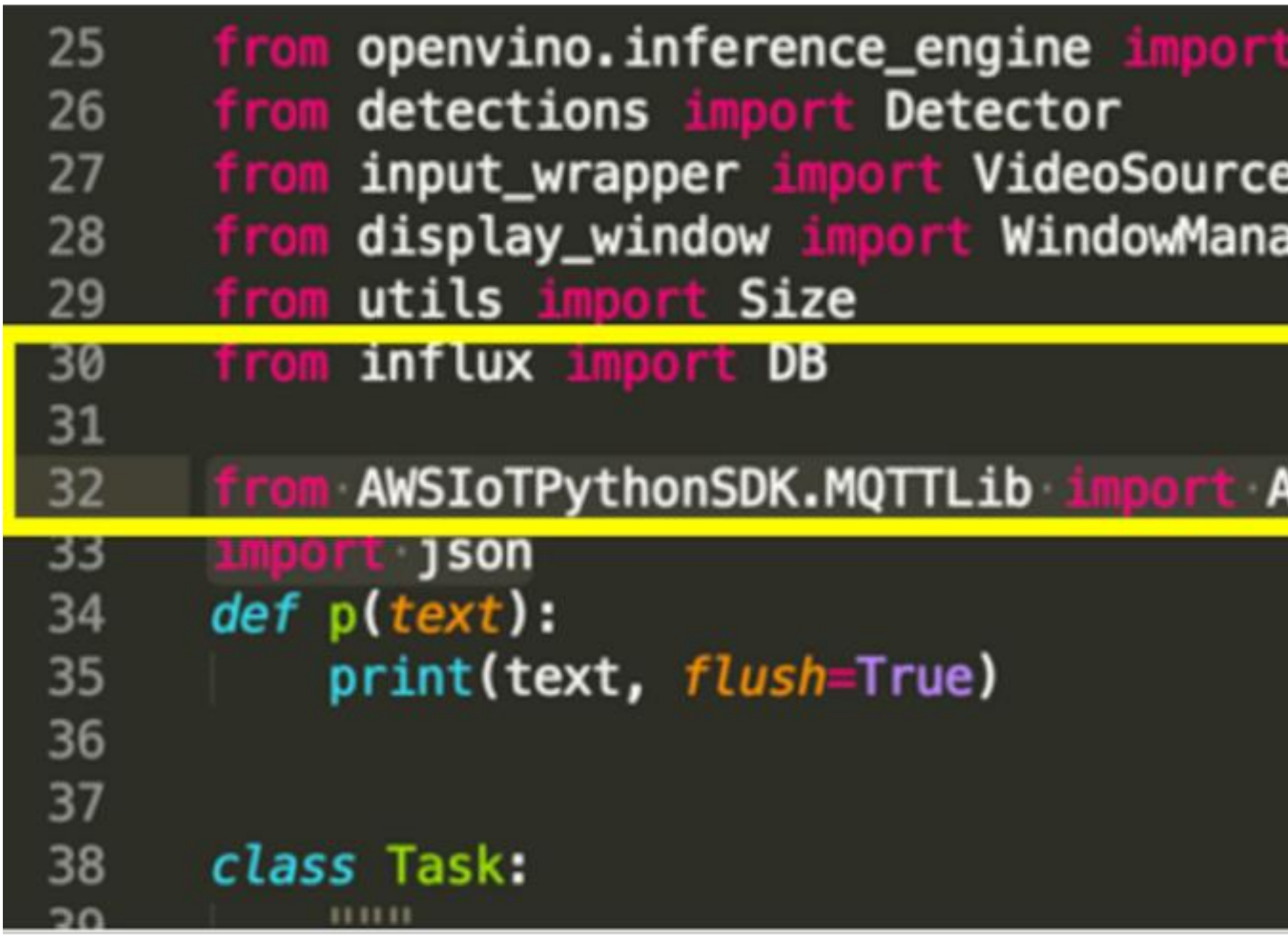
1. **点击以下链接，根据文档说明安装社交距离参考实现方案：** https://software.intel.com/content/www/us/en/develop/articles/multi-camera-monitoring-reference-implementation.html

2. **点击以下链接，按照说明在装有 RI 的机器上安装 AWS IoT python SDK：** https://docs.aws.amazon.com/greengrass/latest/developerguide/IoT-SDK.html

3. **点击以下链接，按照说明在云端配置 AWS 组件并下载证书：** https://docs.aws.amazon.com/greengrass/latest/developerguide/device-group.

html

4. **在“main.py”中修改代码，连接并将数据发送至 AWS cloud。**
   a. 添加导入语句。

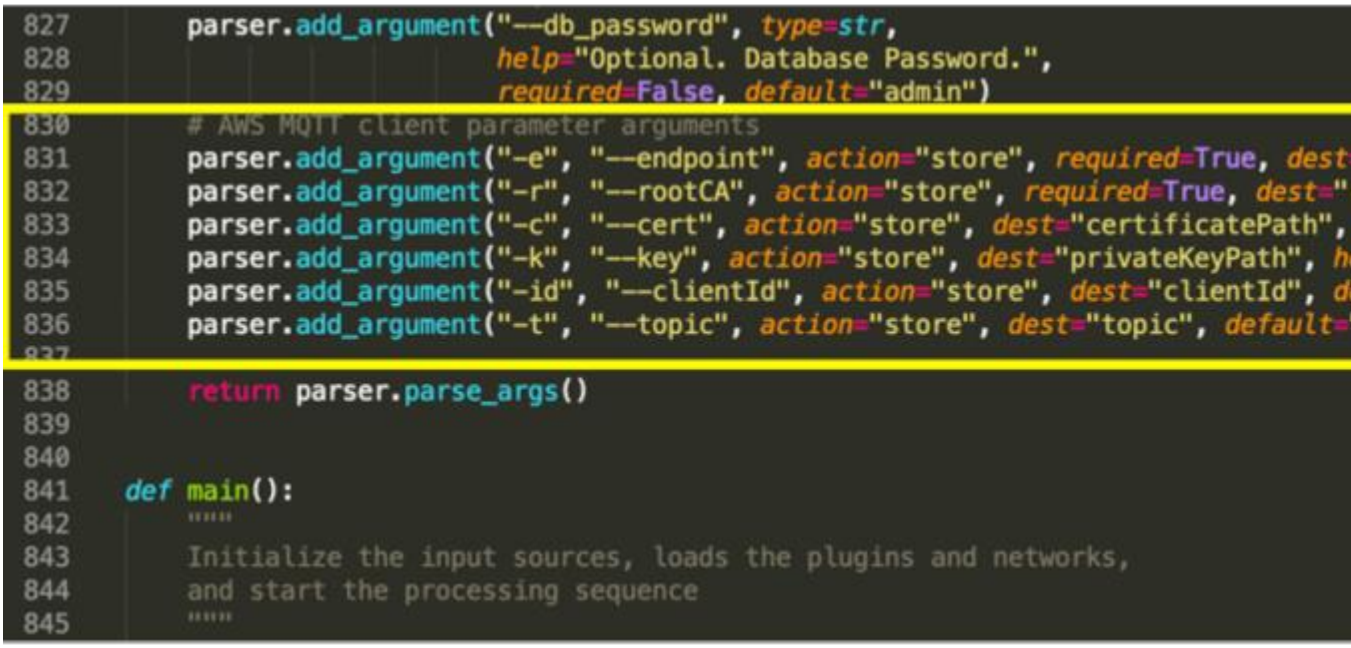


```
from openvino.inference_engine import
from detections import Detector
from input_wrapper import VideoSource
from display_window import WindowMana
from utils import Size
from influx import DB

from AWSIoTPythonSDK.MQTTLib import A
import json
def p(text):
    print(text, flush=True)


class Task:
    """
```

i. 从 AWSIoTPythonSDK.MQTTLib 中导入 AWSIoTMQTTClient

ii. 导入 json

B.添加代码段以获取更多命令行参数，以集成 AWS IoT。



```
    parser.add_argument("--db_password", type=str,
                        help="Optional. Database Password.",
                        required=False, default="admin")
    # AWS MQTT client parameter arguments
    parser.add_argument("-e", "--endpoint", action="store", required=True, dest
    parser.add_argument("-r", "--rootCA", action="store", required=True, dest="
    parser.add_argument("-c", "--cert", action="store", dest="certificatePath",
    parser.add_argument("-k", "--key", action="store", dest="privateKeyPath", h
    parser.add_argument("-id", "--clientId", action="store", dest="clientId", d
    parser.add_argument("-t", "--topic", action="store", dest="topic", default=

    return parser.parse_args()


def main():
    """
    Initialize the input sources, loads the plugins and networks,
    and start the processing sequence
    """
```

i. **# AWS MQTT 客户端参数**

ii. parser.add_argument("-e", "--endpoint", action="store", required=True, dest="host", help="Your AWS IoT custom endpoint")

iii. parser.add_argument("-r", "--rootCA", action="store", required=True, dest="rootCAPath", help="Root CA file path")

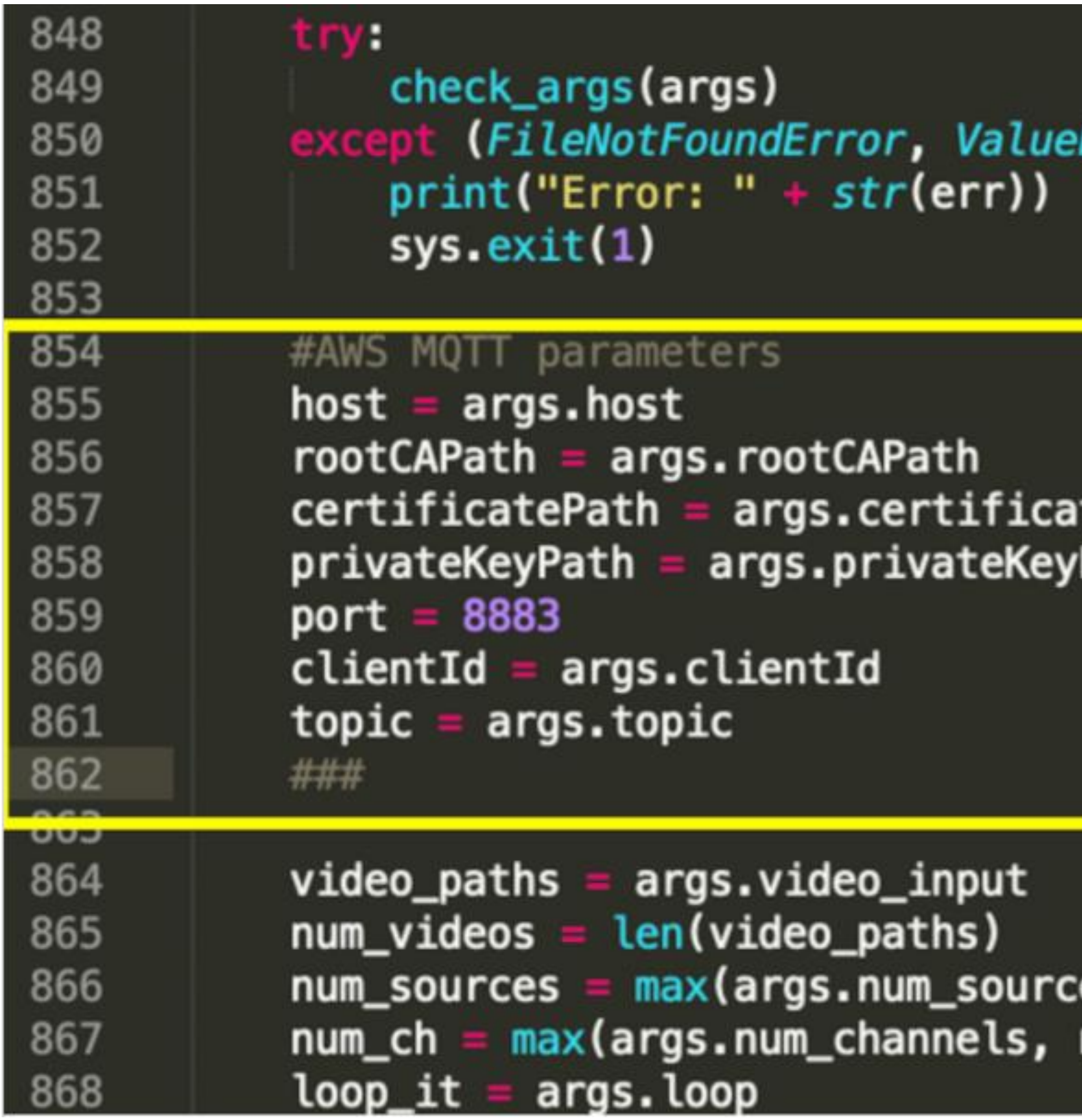
iv. parser.add_argument("-c", "--cert", action="store", dest="certificatePath", help="Certificate file path")

v. parser.add_argument("-k", "--key", action="store", dest="privateKeyPath", help="Private key file path")

vi. parser.add_argument("-id", "--clientId", action="store", dest="clientId", default="basicPubSub",help="Targeted client id")

vii. parser.add_argument("-t", "--topic", action="store", dest="topic", default="sdk/test/Python", help="Targeted topic")

C.使用参数变量设置局部变量。



```
    try:
        check_args(args)
    except (FileNotFoundError, Value
        print("Error: " + str(err))
        sys.exit(1)

    #AWS MQTT parameters
    host = args.host
    rootCAPath = args.rootCAPath
    certificatePath = args.certifica
    privateKeyPath = args.privateKey
    port = 8883
    clientId = args.clientId
    topic = args.topic
    ###

    video_paths = args.video_input
    num_videos = len(video_paths)
    num_sources = max(args.num_sourc
    num_ch = max(args.num_channels,
    loop_it = args.loop
```

i. #AWS MQTT 参数

ii. host = args.host

iii. rootCAPath = args.rootCAPath

iv. certificatePath = args.certificatePath

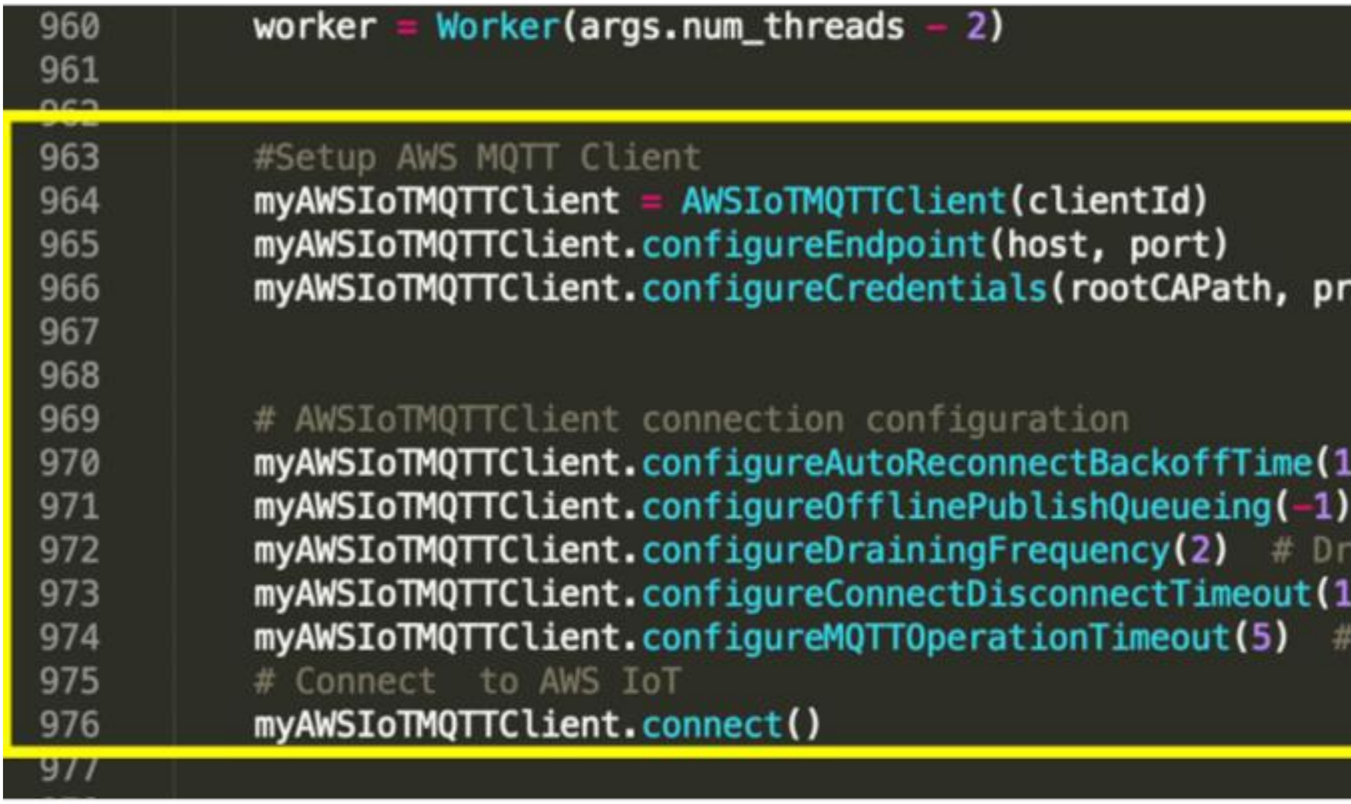
v. privateKeyPath = args.privateKeyPath

vi. port = 8883

vii. clientId = args.clientId

viii. topic = args.topic

D.添加代码段以初始化 MQTT 客户端并设置连接配置。



```
    worker = Worker(args.num_threads - 2)


    #Setup AWS MQTT Client
    myAWSIoTMQTTClient = AWSIoTMQTTClient(clientId)
    myAWSIoTMQTTClient.configureEndpoint(host, port)
    myAWSIoTMQTTClient.configureCredentials(rootCAPath, pr


    # AWSIoTMQTTClient connection configuration
    myAWSIoTMQTTClient.configureAutoReconnectBackoffTime(1
    myAWSIoTMQTTClient.configureOfflinePublishQueueing(-1)
    myAWSIoTMQTTClient.configureDrainingFrequency(2)  # Dr
    myAWSIoTMQTTClient.configureConnectDisconnectTimeout(1
    myAWSIoTMQTTClient.configureMQTTOperationTimeout(5)  #
    # Connect  to AWS IoT
    myAWSIoTMQTTClient.connect()
```

i. #Setup AWS MQTT Client

ii. myAWSIoTMQTTClient = AWSIoTMQTTClient(clientId)

iii. myAWSIoTMQTTClient.configureEndpoint(host, port)

iv. myAWSIoTMQTTClient.configureCredentials(rootCAPath, privateKeyPath, certificatePath)

v. # AWSIoTMQTTClient connection configuration

vi. myAWSIoTMQTTClient.configureAutoReconnectBackoffTime(1, 32, 20)

vii. myAWSIoTMQTTClient.configureOfflinePublishQueueing(-1)   # Infinite offline Publish queueing

viii. myAWSIoTMQTTClient.configureDrainingFrequency(2)   # Draining: 2 Hz

ix. myAWSIoTMQTTClient.configureDrainingFrequency(2)   # Draining: 2 Hz

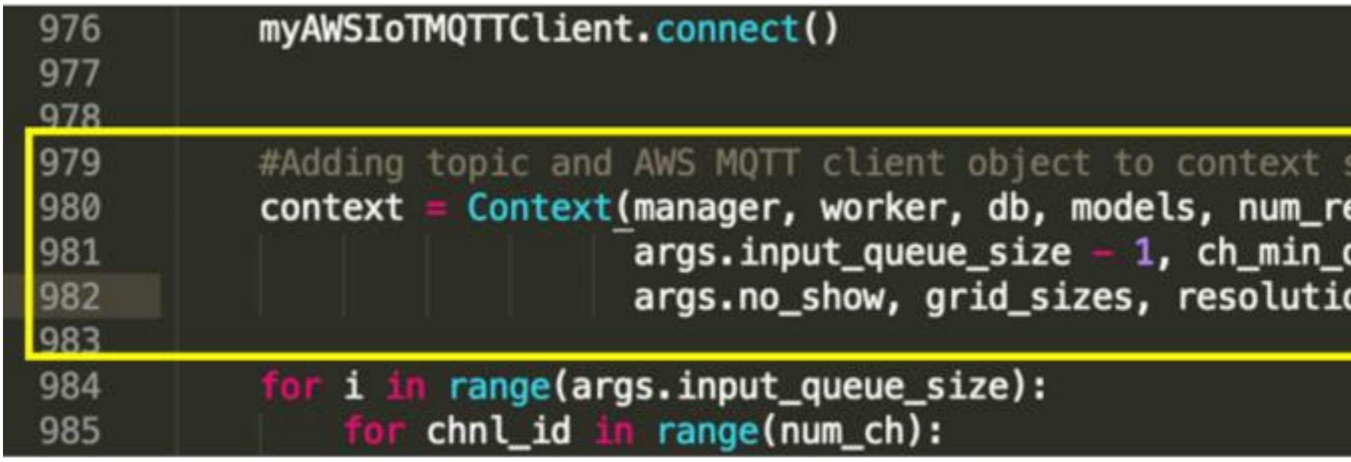
x. myAWSIoTMQTTClient.configureConnectDisconnectTimeout(10)   # 10 sec

xi. myAWSIoTMQTTClient.configureMQTTOperationTimeout(5)   # 5 s

xii.# Connect   to AWS IoT

xiii. myAWSIoTMQTTClient.connect()

E.将主题和 AWS MQTT 客户端对象添加到上下文类。

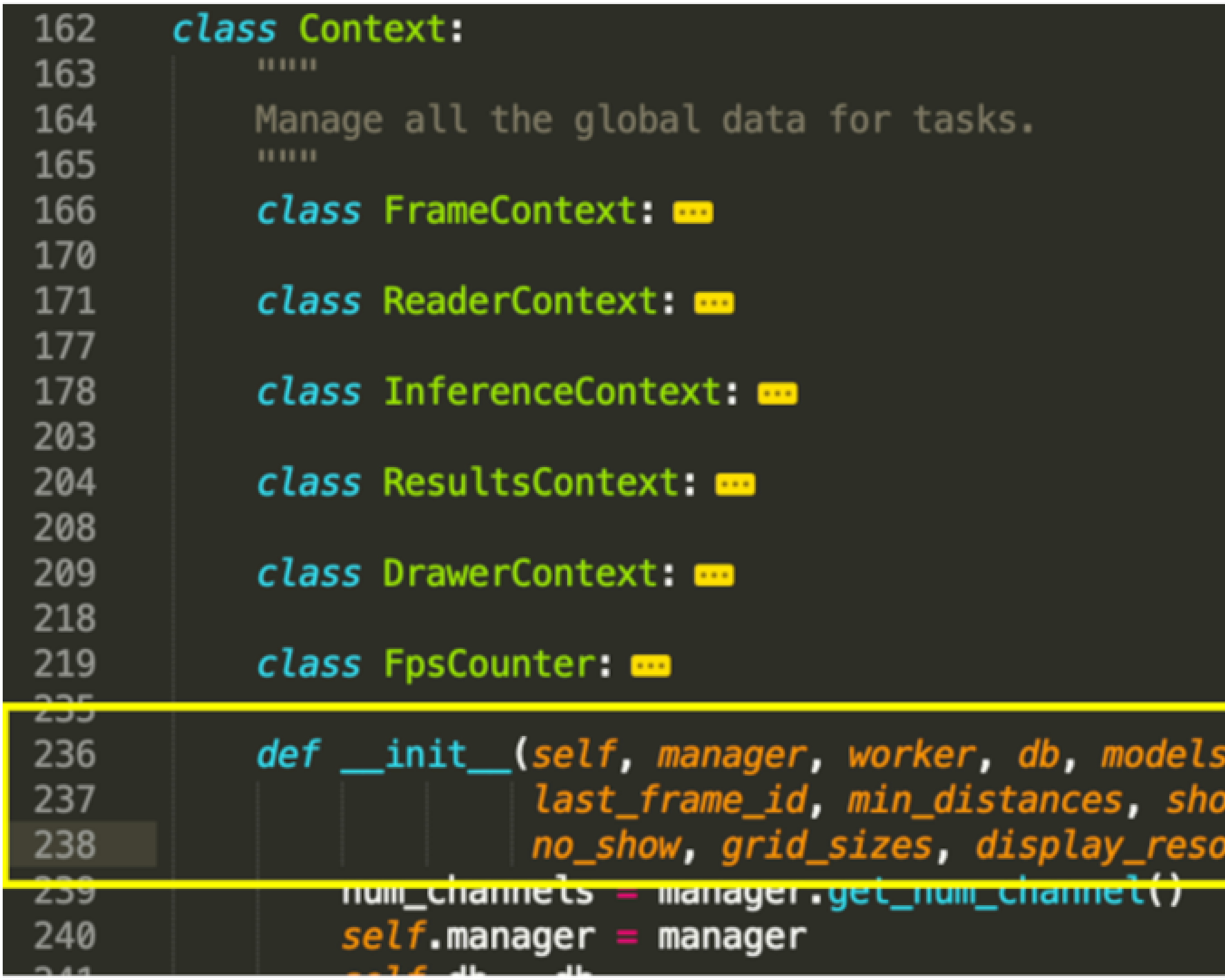


```
    myAWSIoTMQTTClient.connect()


    #Adding topic and AWS MQTT client object to context s
    context = Context(manager, worker, db, models, num_re
                      args.input_queue_size - 1, ch_min_c
                      args.no_show, grid_sizes, resolutio

    for i in range(args.input_queue_size):
        for chnl_id in range(num_ch):
```

i. #Adding topic and AWS MQTT client object to context so it can be shared across the code

ii. context = Context(manager, worker, db, models, num_reqs, args.input_queue_size - 1, ch_min_dist, show_period, args.no_show, grid_sizes, resolution,topic, myAWSIoTMQTTClient)

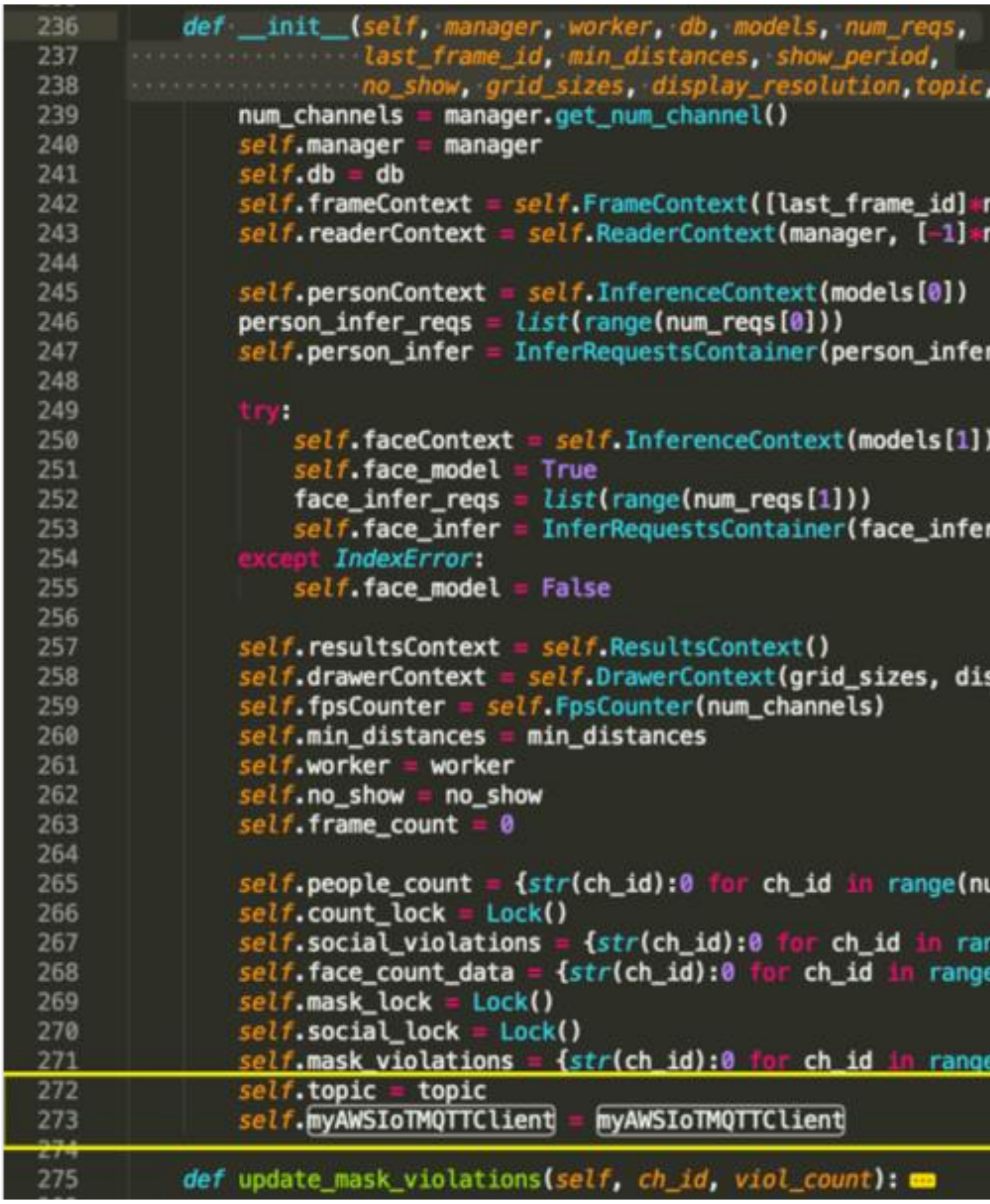
F.将变量添加到上下文类的 init 函数。



```
class Context:
    """
    Manage all the global data for tasks.
    """
    class FrameContext: …

    class ReaderContext: …

    class InferenceContext: …

    class ResultsContext: …

    class DrawerContext: …

    class FpsCounter: …

    def __init__(self, manager, worker, db, models
                 last_frame_id, min_distances, sho
                 no_show, grid_sizes, display_reso
        num_channels = manager.get_num_channel()
        self.manager = manager
        self.db = db
```

i. def __init__(self, manager, worker, db, models, num_reqs, last_frame_id, min_distances, show_period, no_show, grid_sizes, display_resolution,topic,myAWSIoTMQTTClient):

G.使用传递到上下文类的 init 函数的新值初始化局部变量

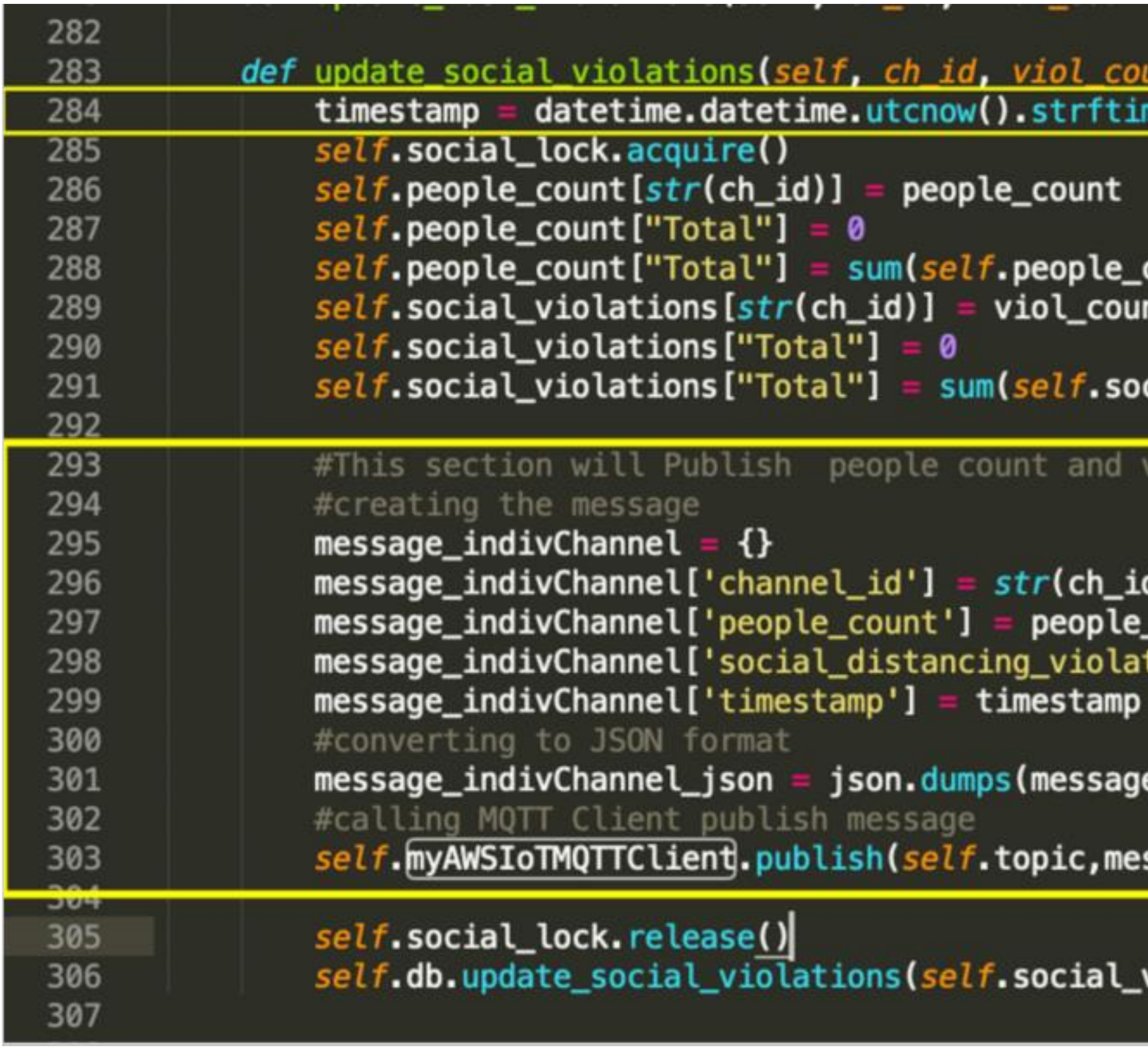


```
    def __init__(self, manager, worker, db, models, num_reqs,
                 last_frame_id, min_distances, show_period,
                 no_show, grid_sizes, display_resolution,topic,
        num_channels = manager.get_num_channel()
        self.manager = manager
        self.db = db
        self.frameContext = self.FrameContext([last_frame_id]*n
        self.readerContext = self.ReaderContext(manager, [-1]*n

        self.personContext = self.InferenceContext(models[0])
        person_infer_reqs = list(range(num_reqs[0]))
        self.person_infer = InferRequestsContainer(person_infe

        try:
            self.faceContext = self.InferenceContext(models[1])
            self.face_model = True
            face_infer_reqs = list(range(num_reqs[1]))
            self.face_infer = InferRequestsContainer(face_infe
        except IndexError:
            self.face_model = False

        self.resultsContext = self.ResultsContext()
        self.drawerContext = self.DrawerContext(grid_sizes, dis
        self.fpsCounter = self.FpsCounter(num_channels)
        self.min_distances = min_distances
        self.worker = worker
        self.no_show = no_show
        self.frame_count = 0

        self.people_count = {str(ch_id):0 for ch_id in range(nu
        self.count_lock = Lock()
        self.social_violations = {str(ch_id):0 for ch_id in ran
        self.face_count_data = {str(ch_id):0 for ch_id in range
        self.mask_lock = Lock()
        self.social_lock = Lock()
        self.mask_violations = {str(ch_id):0 for ch_id in range
        self.topic = topic
        self.myAWSIoTMQTTClient = myAWSIoTMQTTClient

    def update_mask_violations(self, ch_id, viol_count):
```

i. self.topic = topic

ii. self.myAWSIoTMQTTClient = myAWSIoTMQTTClient

H.修改 update_social_violations 函数，将数据发送至 AWS IoT



```
    def update_social_violations(self, ch_id, viol_cou
        timestamp = datetime.datetime.utcnow().strftim
        self.social_lock.acquire()
        self.people_count[str(ch_id)] = people_count
        self.people_count["Total"] = 0
        self.people_count["Total"] = sum(self.people_
        self.social_violations[str(ch_id)] = viol_cou
        self.social_violations["Total"] = 0
        self.social_violations["Total"] = sum(self.soc

        #This section will Publish  people count and v
        #creating the message
        message_indivChannel = {}
        message_indivChannel['channel_id'] = str(ch_i
        message_indivChannel['people_count'] = people_
        message_indivChannel['social_distancing_violat
        message_indivChannel['timestamp'] = timestamp
        #converting to JSON format
        message_indivChannel_json = json.dumps(message
        #calling MQTT Client publish message
        self.myAWSIoTMQTTClient.publish(self.topic,mes

        self.social_lock.release()
        self.db.update_social_violations(self.social_
```

i. 以特定格式添加时间戳，将其发送至 AWS TimeStream 数据库

1. timestamp = datetime.datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S.%f")[:-3]

ii.添加代码段以创建 MQTT 消息并发布到 AWS IoT

1.#This section will Publish people count and violations to AWS IoT main topic

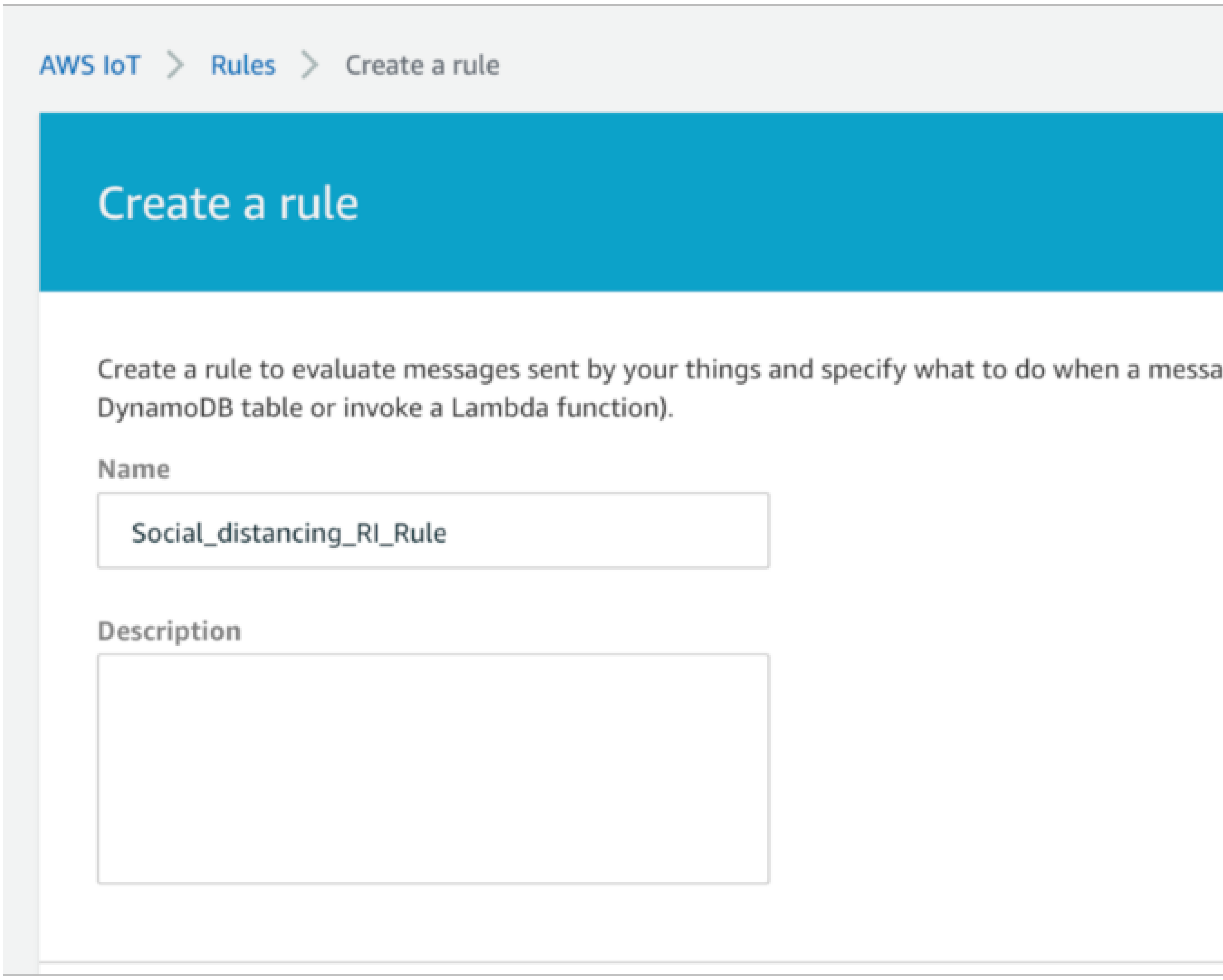
2.#creating the message

3. message_indivChannel = {}

```
message_indivChannel['channel_id'] = str(ch_id)
message_indivChannel['people_count'] = people_count
message_indivChannel['social_distancing_violation'] = viol_count
message_indivChannel['timestamp'] = timestamp
#converting to JSON format
message_indivChannel_json = json.dumps(message_indivChannel)
#calling MQTT Client publish message
self.myAWSIoTMQTTClient.publish(self.topic,message_indivChannel_json, 1)
```

5.配置 AWS IoT，以将数据存储到 Timestream 数据库。

a. 添加新规则。



Create a rule

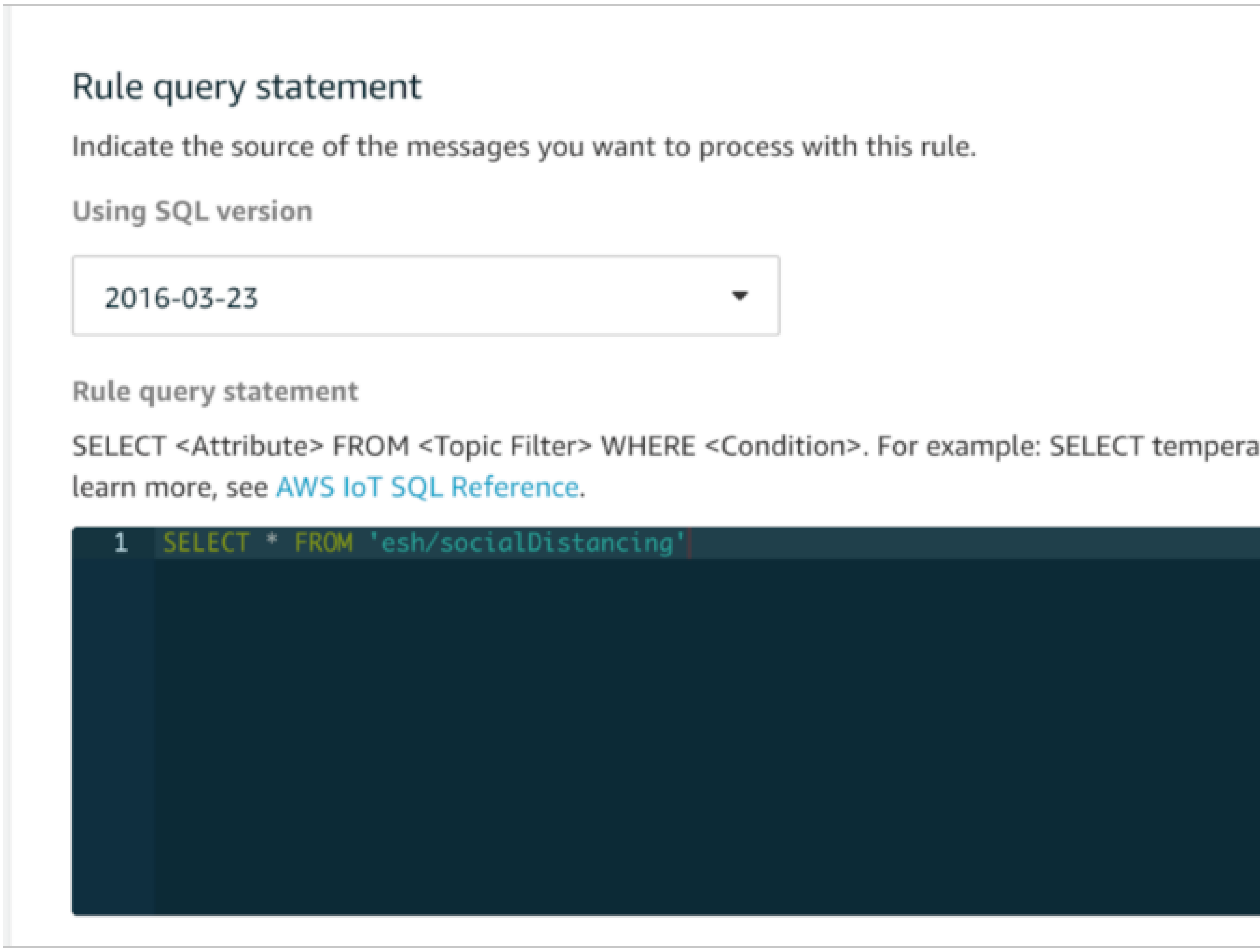
Create a rule to evaluate messages sent by your things and specify what to do when a messa
DynamoDB table or invoke a Lambda function).

Name

Social_distancing_RI_Rule

Description

b.添加规则查询语句。



Rule query statement

Indicate the source of the messages you want to process with this rule.

Using SQL version

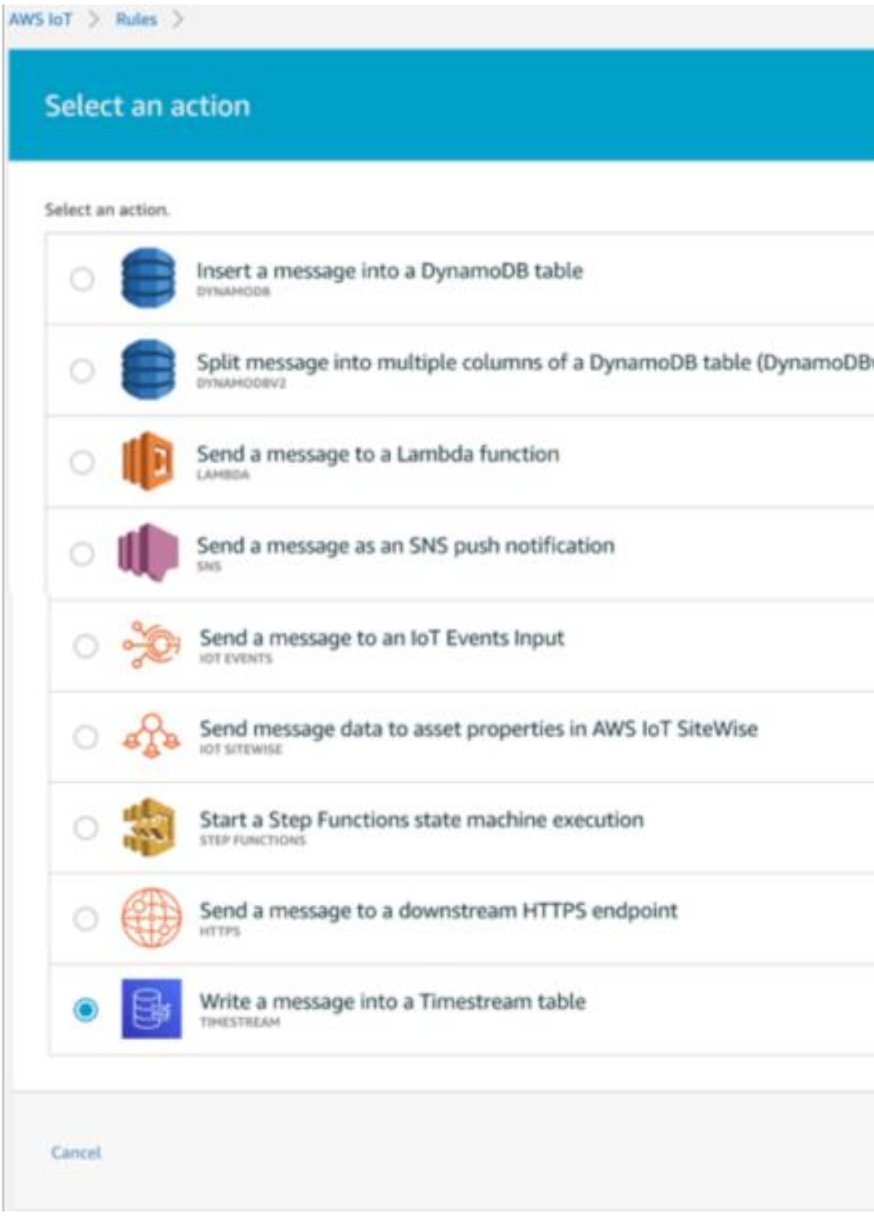
2016-03-23

Rule query statement

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT tempera
learn more, see AWS IoT SQL Reference.

```
SELECT * FROM 'esh/socialDistancing'
```

C.添加操作，选择 Timestream 表。



Select an action

Select an action.

○ Insert a message into a DynamoDB table
DYNAMODB

○ Split message into multiple columns of a DynamoDB table (DynamoDB
DYNAMODBV2

○ Send a message to a Lambda function
LAMBDA

○ Send a message as an SNS push notification
SNS

○ Send a message to an IoT Events Input
IOT EVENTS

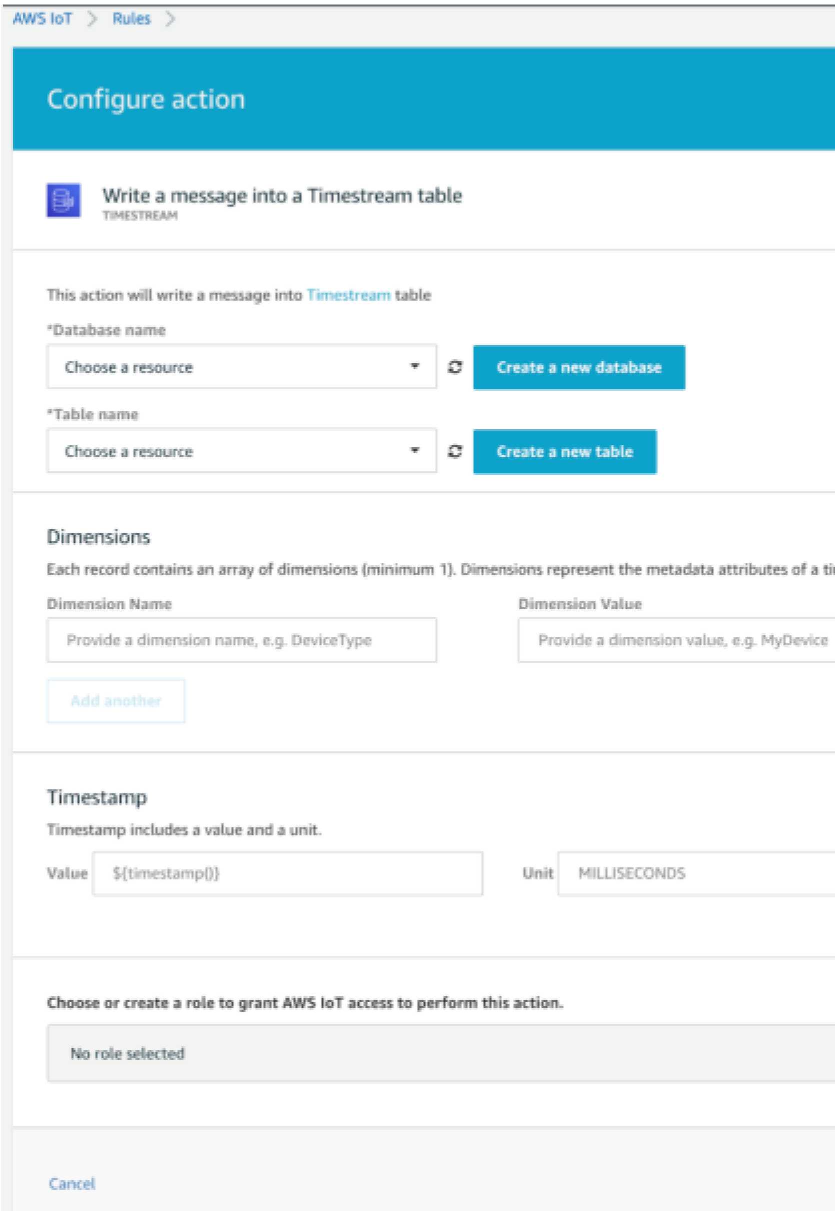
○ Send message data to asset properties in AWS IoT SiteWise
IOT SITEWISE

○ Start a Step Functions state machine execution
STEP FUNCTIONS

○ Send a message to a downstream HTTPS endpoint
HTTPS

● Write a message into a Timestream table
TIMESTREAM

D.配置 Timestream 操作



Configure action

Write a message into a Timestream table

TIMESTREAM

This action will write a message into Timestream table

*Database name

Choose a resource

*Table name

Choose a resource

Dimensions

Each record contains an array of dimensions (minimum 1). Dimensions represent the metadata attributes of a ti

Dimension Name: Provide a dimension name, e.g. DeviceType

Dimension Value: Provide a dimension value, e.g. MyDevice

Timestamp

Timestamp includes a value and a unit.

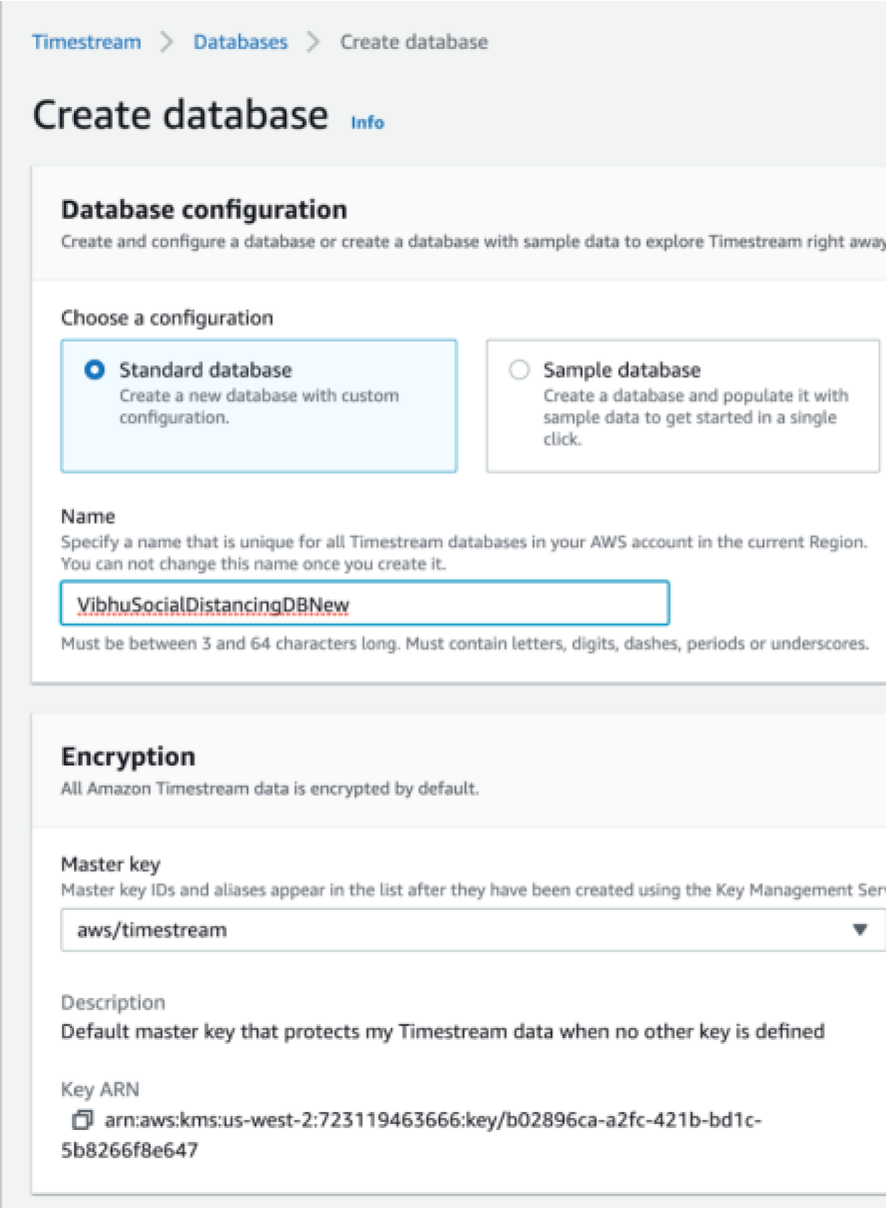
Value: ${timestamp()}

Unit: MILLISECONDS

Choose or create a role to grant AWS IoT access to perform this action.

No role selected

e.创建一个新的数据库



Create database

Database configuration

Create and configure a database or create a database with sample data to explore Timestream right away

Choose a configuration

● Standard database
Create a new database with custom configuration.

○ Sample database
Create a database and populate it with sample data to get started in a single click.

Name

Specify a name that is unique for all Timestream databases in your AWS account in the current Region. You can not change this name once you create it.

VibhuSocialDistancingDBNew

Must be between 3 and 64 characters long. Must contain letters, digits, dashes, periods or underscores.

Encryption

All Amazon Timestream data is encrypted by default.

Master key

Master key IDs and aliases appear in the list after they have been created using the Key Management Ser

aws/timestream ▼

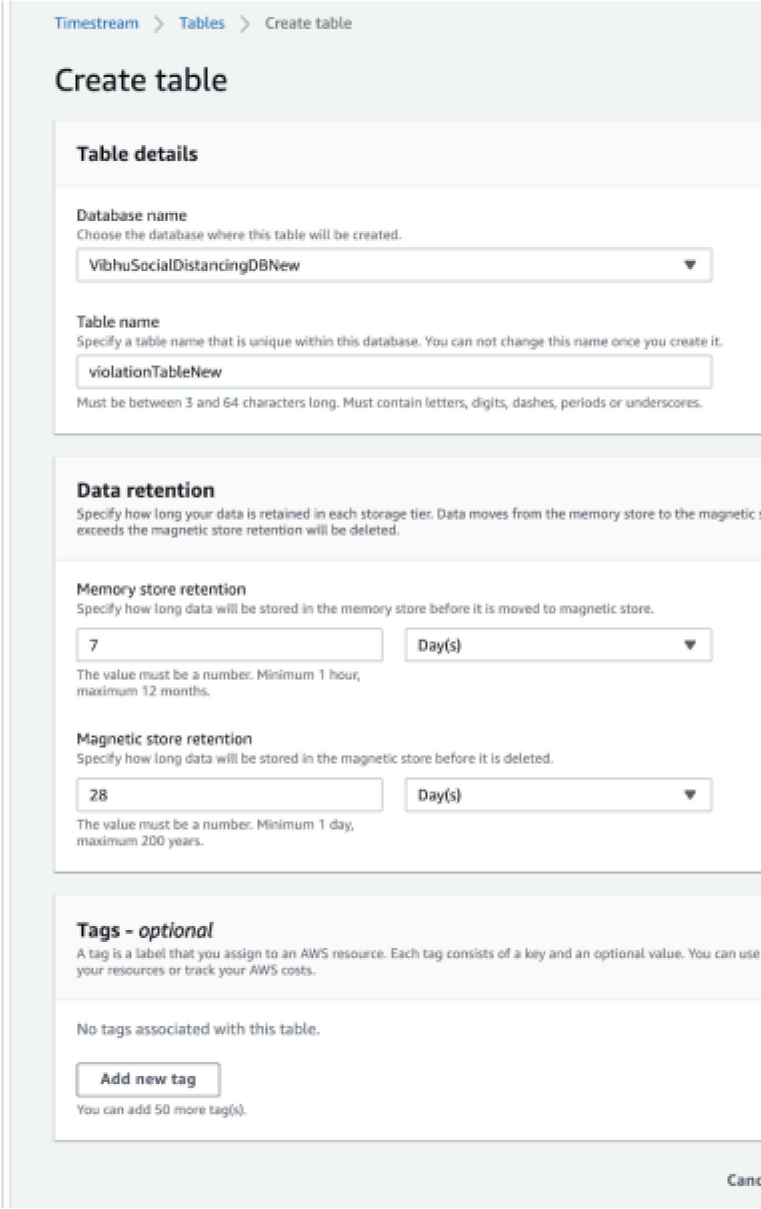
Description

Default master key that protects my Timestream data when no other key is defined

Key ARN

arn:aws:kms:us-west-2:723119463666:key/b02896ca-a2fc-421b-bd1c-5b8266f8e647

F.创建一个新表。



Timestream > Tables > Create table

Create table

Table details

Database name
Choose the database where this table will be created.
VibhuSocialDistancingDBNew

Table name
Specify a table name that is unique within this database. You can not change this name once you create it.
violationTableNew
Must be between 3 and 64 characters long. Must contain letters, digits, dashes, periods or underscores.

Data retention
Specify how long your data is retained in each storage tier. Data moves from the memory store to the magnetic s
exceeds the magnetic store retention will be deleted.

Memory store retention
Specify how long data will be stored in the memory store before it is moved to magnetic store.
7 Day(s)
The value must be a number. Minimum 1 hour, maximum 12 months.

Magnetic store retention
Specify how long data will be stored in the magnetic store before it is deleted.
28 Day(s)
The value must be a number. Minimum 1 day, maximum 200 years.

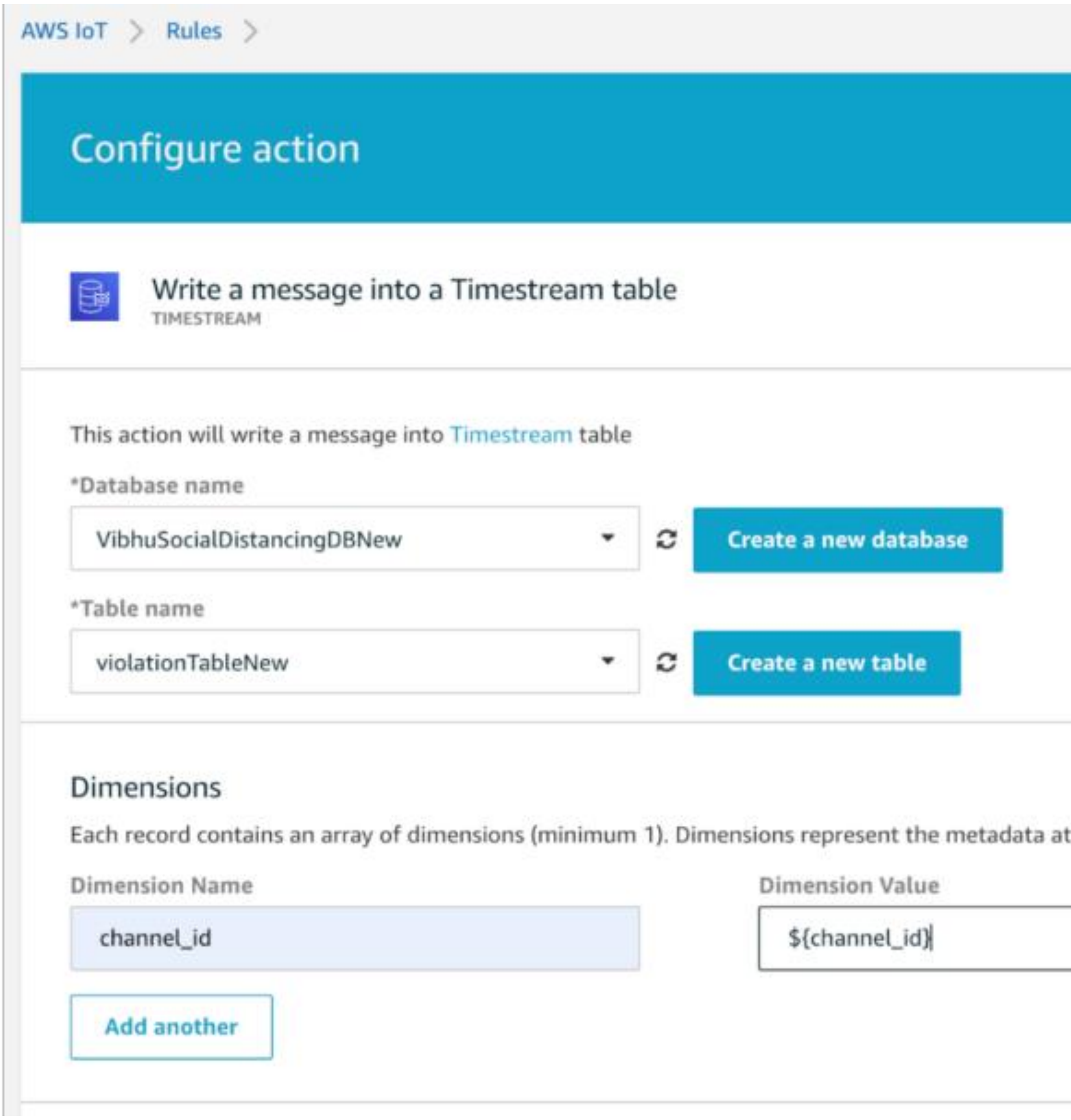
Tags - *optional*
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use
your resources or track your AWS costs.

No tags associated with this table.

You can add 50 more tag(s).

g. 将尺寸设置为 channel_ id

注意，尺寸不能是“整数”



Configure action

Write a message into a Timestream table

TIMESTREAM

This action will write a message into Timestream table

*Database name

VibhuSocialDistancingDBNew

*Table name

violationTableNew

Dimensions

Each record contains an array of dimensions (minimum 1). Dimensions represent the metadata at

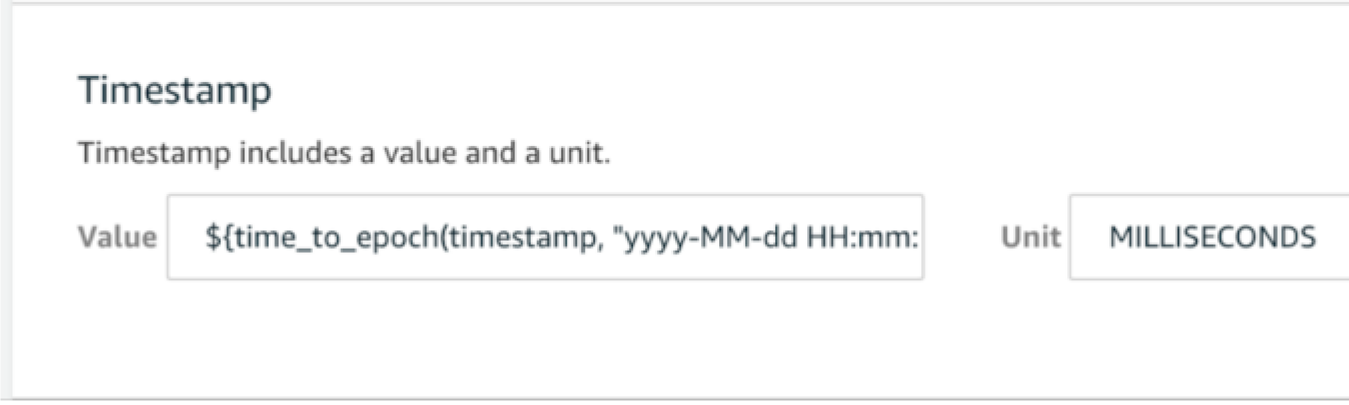
Dimension Name: channel_id

Dimension Value: ${channel_id}

H.通过解析 MQTT 有效负载中的数据来设置时间戳

1.Value - ${time_to_epoch(timestamp, "yyyy-MM-dd HH:mm:ss.SSS")}

2.Unit - MILLISECONDS



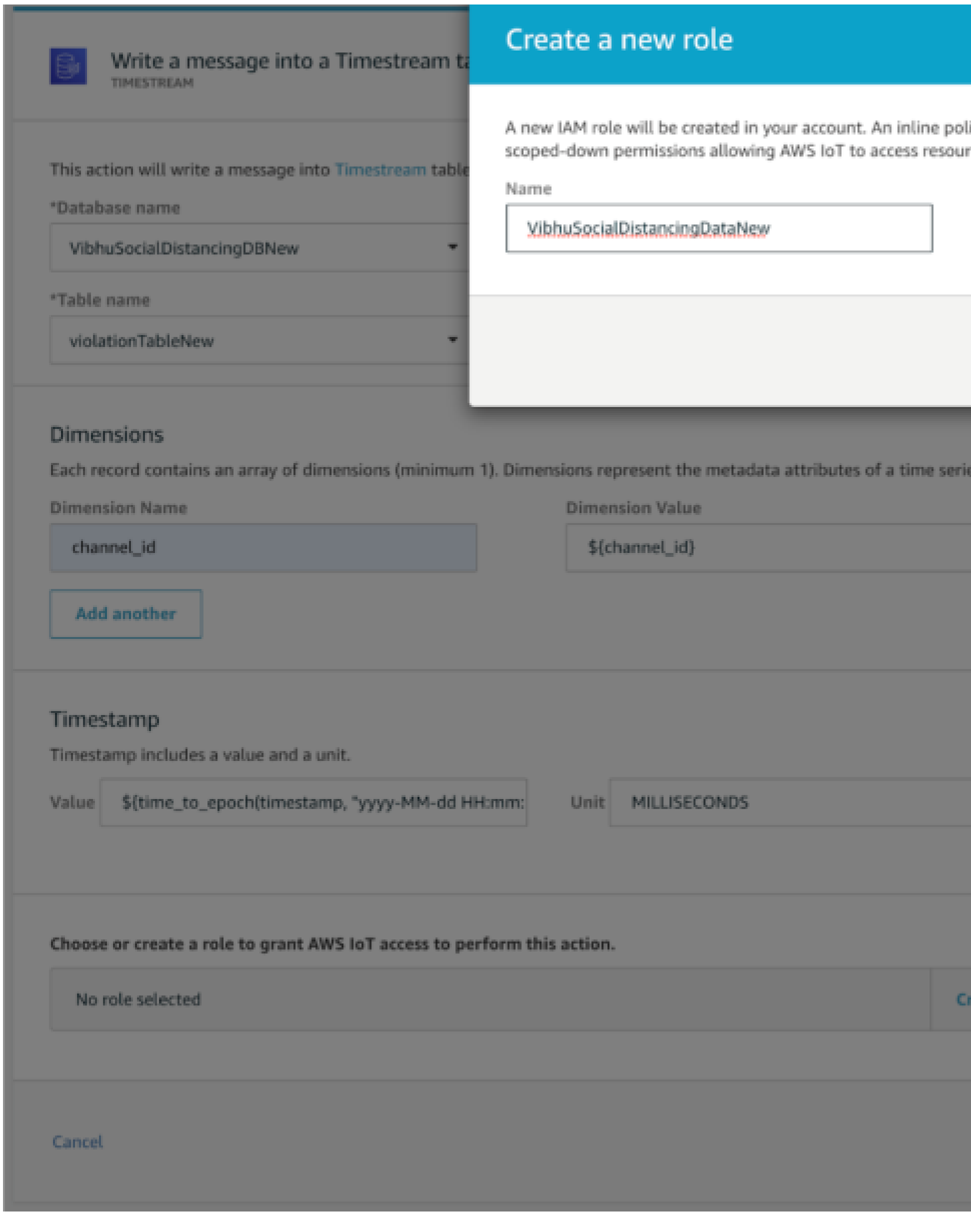
Timestamp

Timestamp includes a value and a unit.

Value ${time_to_epoch(timestamp, "yyyy-MM-dd HH:mm:

Unit MILLISECONDS

i. 创建角色。



Write a message into a Timestream ta

TIMESTREAM

This action will write a message into Timestream table

*Database name

VibhuSocialDistancingDBNew

*Table name

violationTableNew

Dimensions

Each record contains an array of dimensions (minimum 1). Dimensions represent the metadata attributes of a time serie

Dimension Name	Dimension Value
channel_id	${channel_id}

Timestamp

Timestamp includes a value and a unit.

Value ${time_to_epoch(timestamp, "yyyy-MM-dd HH:mm: Unit MILLISECONDS

Choose or create a role to grant AWS IoT access to perform this action.

No role selected

Create a new role

A new IAM role will be created in your account. An inline poli scoped-down permissions allowing AWS IoT to access resour

Name

VibhuSocialDistancingDataNew

6.设置 Grafana

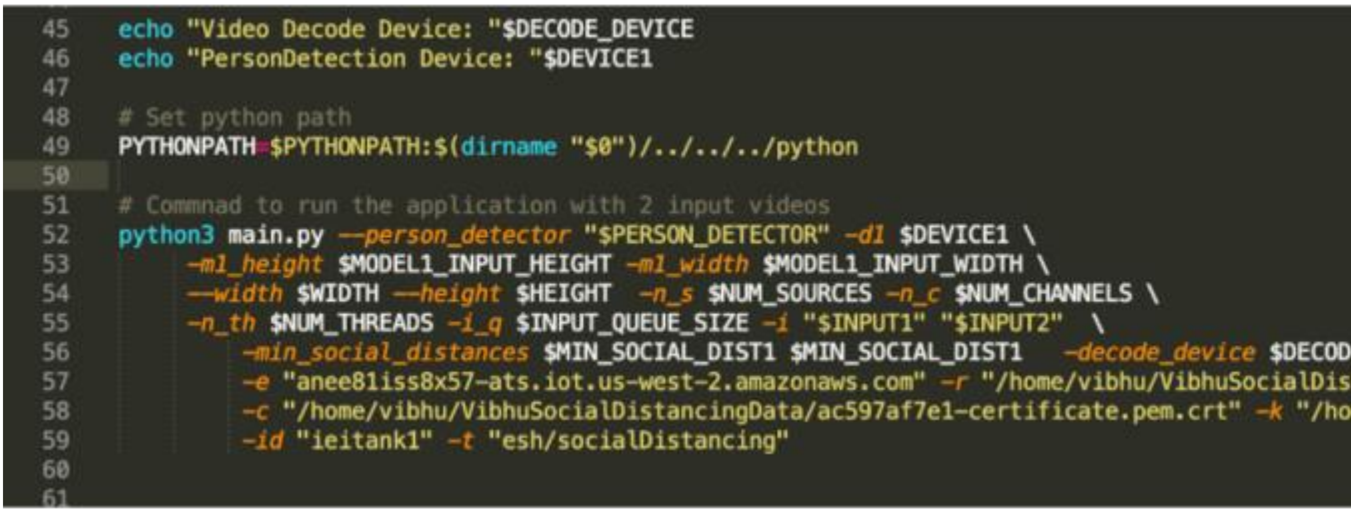
a.将主机上的 Grafana 升级到最新的机器中。

B.添加 AWS timestream 插件。

C.使用您的凭证配置 AWS 插件。

d.配置您的仪表板。

7.更新 run.sh 文件



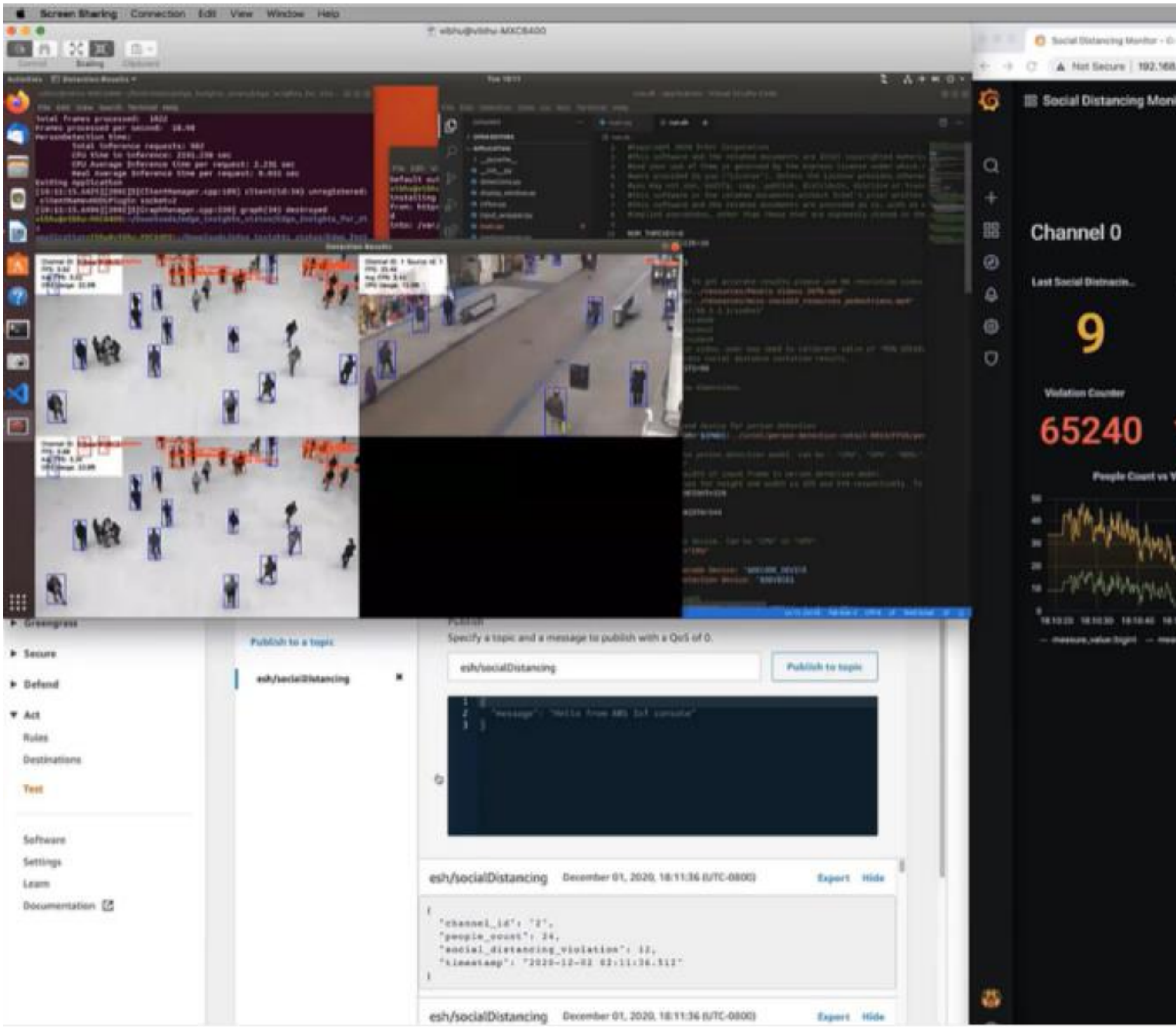
```
echo "Video Decode Device: "$DECODE_DEVICE
echo "PersonDetection Device: "$DEVICE1

# Set python path
PYTHONPATH=$PYTHONPATH:$(dirname "$0")/../../../python

# Commnad to run the application with 2 input videos
python3 main.py --person_detector "$PERSON_DETECTOR" -d1 $DEVICE1 \
    -m1_height $MODEL1_INPUT_HEIGHT -m1_width $MODEL1_INPUT_WIDTH \
    --width $WIDTH --height $HEIGHT  -n_s $NUM_SOURCES -n_c $NUM_CHANNELS \
    -n_th $NUM_THREADS -i_q $INPUT_QUEUE_SIZE -i "$INPUT1" "$INPUT2"  \
        -min_social_distances $MIN_SOCIAL_DIST1 $MIN_SOCIAL_DIST1   -decode_device $DECOD
        -e "anee81iss8x57-ats.iot.us-west-2.amazonaws.com" -r "/home/vibhu/VibhuSocialDis
        -c "/home/vibhu/VibhuSocialDistancingData/ac597af7e1-certificate.pem.crt" -k "/ho
        -id "ieitank1" -t "esh/socialDistancing"
```

```
python3 main.py --person_detector "$PERSON_DETECTOR" -d1 $DEVICE1 \

 -m1_height $MODEL1_INPUT_HEIGHT -m1_width $MODEL1_INPUT_WIDTH
\
--width $WIDTH --height $HEIGHT   -n_s $NUM_SOURCES -n_c
$NUM_CHANNELS \
-n_th $NUM_THREADS -i_q $INPUT_QUEUE_SIZE -i "$INPUT1" "$INPUT2"   \
-min_social_distances $MIN_SOCIAL_DIST1
$MIN_SOCIAL_DIST1    -decode_device $DECODE_DEVICE \
-e "anee81iss8x57-ats.iot.us-west-2.amazonaws.com" -r
"/home/vibhu/VibhuSocialDistancingData/AmazonRootCA1.pem" \
-c "/home/vibhu/VibhuSocialDistancingData/ac597af7e1-certificate.pem.crt"
```

*-k "/home/vibhu/VibhuSocialDistancingData/ac597af7e1-private.pem.key" \
-id "ieitank1" -t "esh/socialDistancing"*

8.运行带有示例视频的应用。

违反社交距离规定的行为将在视频中被标记出来，用户可通过仪表板监控性能。



```
Total frames processed: 1822
Frames processed per second: 18.98
PersonDetection time:
        Total Inference requests: 902
        CPU time to inference: 2101.238 sec
        CPU Average Inference time per request: 2.231 sec
        Real Average Inference time per request: 0.031 sec
Exiting Application
```

Specify a topic and a message to publish with a QoS of 0.

esh/socialDistancing

```
{
  "message": "Hello from AWS IoT console"
}
```

esh/socialDistancing December 01, 2020, 18:11:36 (UTC-0800)

```
{
  "channel_id": "2",
  "people_count": 24,
  "social_distancing_violation": 12,
  "timestamp": "2020-12-02 02:11:36.512"
}
```

esh/socialDistancing December 01, 2020, 18:11:36 (UTC-0800)

更多用例和软件产品

开发人员渴望创建定制的 AI 解决方案以解决实际问题。发现问题后，需要加快上市时间、降低开发成本并借助强大的生态系统进行扩展。为实现该目的，英特尔在英特尔® 边缘软件中心上为开发人员提供了支持部署的可复用容器化软件包和用例。开发人员可以找到参考实现方案，包括大量边缘到云端 AI 应用的教程、示例代码和文档。

更多资源

点击以下链接，根据文档说明安装社交距离参考实现方案：

https://software.intel.com/content/www/us/en/develop/articles/multi-camera-monitoring-reference-implementation.html

点击以下链接，按照说明在装有 RI 的机器上安装 AWS IoT python SDK

https://docs.aws.amazon.com/greengrass/latest/developerguide/IoT-SDK.html

点击以下链接，按照说明在云端配置 AWS 组件并下载证书：

https://docs.aws.amazon.com/greengrass/latest/developerguide/device-group.html

在“main.py”中修改代码，连接并将数据发送至 AWS cloud。

通知和免责声明

英特尔技术可能需要支持的硬件、特定软件或服务激活。

没有任何产品或组件是绝对安全的。

您的成本或结果可能有所差异。