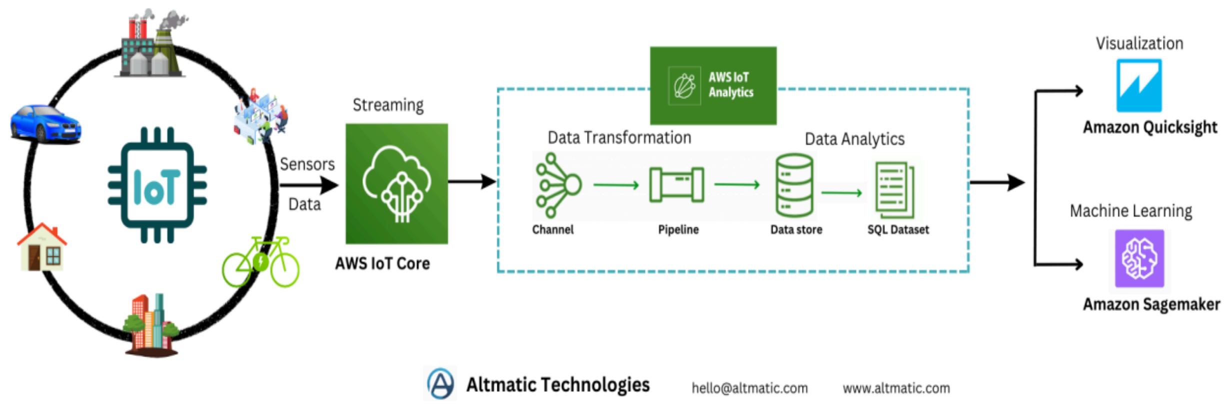
[1]Analyze IoT Data with AWS IoT Analytics

AWS IoT Analytics automates the steps that are required to analyze data from Internet of Things (IoT) devices. It filters, transforms, and enriches IoT data before storing it in a time-series data store for analysis. You can set up the service to collect only the data you need from your devices and apply mathematical transforms to process the data. You can also enrich the data with device-specific metadata—such as device type and location—before you store it. Then, you can analyze your data by running queries using the built-in structured query language (SQL) query engine. You can also perform more complex analytics and machine learning inference. AWS IoT Analytics enables data visualization through integration with Amazon QuickSight.



You work with AWS IoT Analytics by using the following components:

- Channels – Channels collect and archive the raw messages from your IoT devices.
- Pipelines – Pipelines consume messages from channels. You can process the messages in the pipeline before storing them in a data store.
- Data store – A data store is a scalable repository for the messages coming from your IoT devices. You can query and filter the messages in the repository. A single data store can receive input from multiple channels and pipelines.
- Dataset – You retrieve data from a data store by creating a dataset. You can either create an SQL dataset, or a container dataset. In this lab, you will create an SQL dataset.

Objectives

After completing this lab, you will be able to:

- Access the AWS IoT Analytics Service in the AWS Management Console
- Create an AWS IoT Analytics channel

[1] Vijay Kumar Data Engineer

- Create an AWS IoT Analytics data store
- Create an AWS IoT Analytics pipeline
- Create an AWS IoT Core rule
- Query an AWS IoT Analytics data store

Prerequisites

This lab requires:

- Access to a notebook computer with Wi-Fi and Microsoft Windows, macOS, or Linux (Ubuntu, SuSE, or Red Hat)
- For Microsoft Windows users: Administrator access to the computer
- An internet browser such as Chrome, Firefox, or IE9 (previous versions of Internet Explorer are not supported)

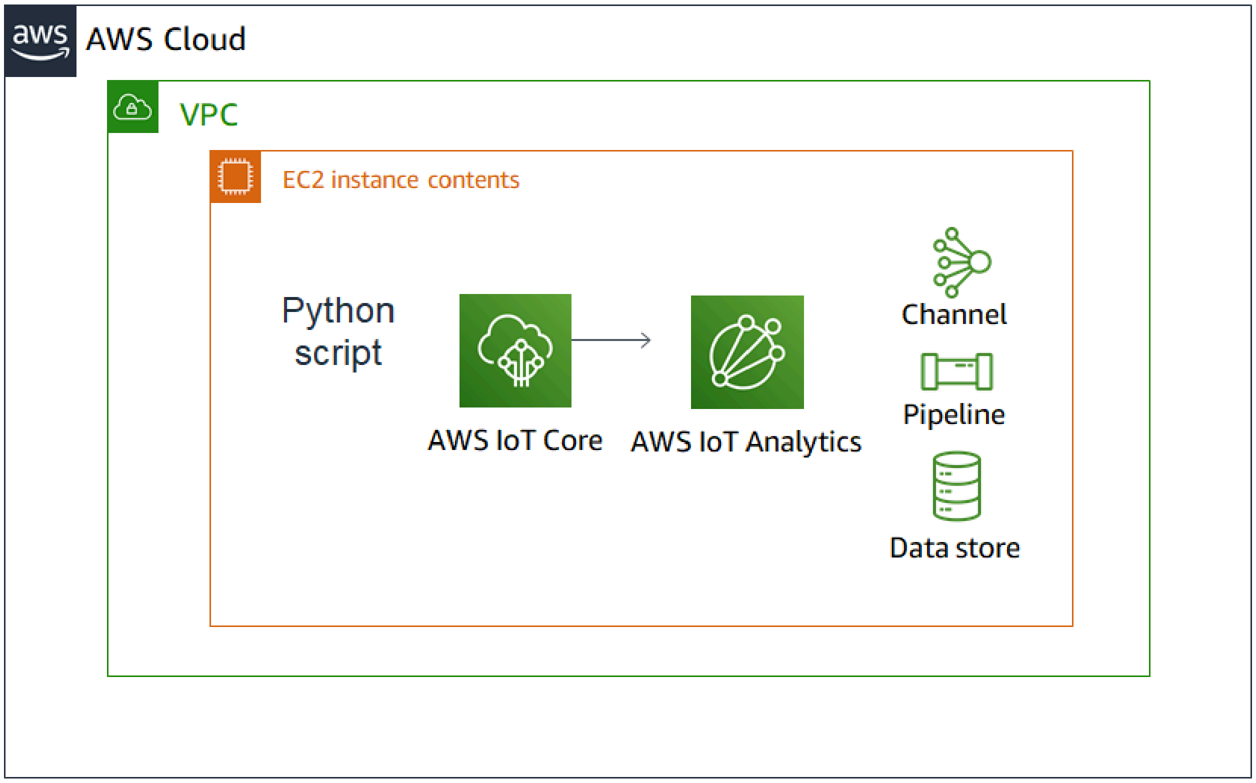
Duration

This lab requires **60** minutes to complete. The lab will remain active for **120** minutes to allow extra time to complete the challenge question.

Scenario summary

In this lab, you are working for an urban planning agency of a major city. The city installed sensors to detect light and humidity levels, as well as temperatures. They intend to use the data to learn more about how vegetation can affect the temperature in the city. The data was collected from sensor readings every 5 minutes from five different locations. In the lab, you import data from these sensors. In an actual IoT implementation, the data sensors would send data directly to your AWS IoT Analytics and AWS IoT Core implementations.

To read more about AWS IoT Analytics, see What is AWS IoT Analytics.



Task 1: Create a channel

A *channel* collects and archives raw, unprocessed message data before it publishes the data to a pipeline.

To create a *channel*:

5. On the AWS Management Console, on the **Services** menu, choose **Services**.
6. From the list of services, choose **IoT Analytics**.
7. In the navigation pane, choose **Channels**.
8. Choose **Create a channel**.
9. In the **Channel ID** box, enter MyChannel.
   As your channel receives data, you will want to store it. AWS IoT Analytics can create and manage this data store for you, or you can store it in an S3 bucket you manage. For this lab, you will choose AWS IoT Analytics to manage the storage.
10. Select **Service-managed store**.
11. Choose **Next**.

12. On the **Set ID, source, and data retention period** page, choose **Create Channel**.

Task 2: Create a data store

Pipelines store their processed messages in a data store. A data store is not a database. It is a scalable repository of your messages you can query. You can have multiple data stores for messages that come from different devices or locations, or are filtered by message attributes, depending on your pipeline configuration and requirements.

To create a data store:

13. In the navigation pane, choose **Data stores**.
14. On the **Data stores** page, choose **Create a data store**.
15. In the **ID** field, enter my_datastore.
16. Select **Service-managed store**.
17. For the data retention period, accept the default value of *Indefinitely*.
18. Choose **Create data store**.

Task 3: Create a pipeline

A pipeline consumes messages from one or more channels. It enables you to process the messages before you store them in a data store. In this exercise, you will upload messages through an AWS IoT Core topic.

To create a pipeline:

19. In the navigation pane, choose **Pipelines**.
20. Choose **Create a pipeline**.
21. In the **Pipeline ID** field, enter my_pipeline.
22. To select the **Pipeline source**, choose **Edit** .
23. Choose **my_channel**.
24. Choose **Next**.
    AWS IoT Analytics attempts to retrieve attributes from messages in your channel. Because you have not created any messages, you must create the attributes manually.
25. For the first attribute, enter itemid.
26. Choose **Add new**.

27. For the next attribute, enter Date.
28. Choose **Add new**.
29. Choose **Next**.
30. On the **Enrich, transform, and filter messages** page, choose **Next**.
31. On the **Save your processed messages in a data store** page, choose **Edit**.
32. Choose the *my_datastore* data store that you created earlier, and then choose **Create pipeline**.

Task 4: Create an AWS IoT Core rule

AWS IoT Analytics integrates with the AWS IoT Core service through a *message broker*. IoT devices send their data to a *topic*. The messages are sent using the Message Queuing Telemetry Transport (MQTT) protocol. The message broker then sends these messages to client applications that registered to receive the message. Sending messages to the message broker is known as *publishing*, and registering to receive messages for a topic is known as *subscribing*. In this task, you will use a Python script to simulate publishing from IoT devices. To do this task, you will:

- Create an AWS IoT Core rule that will publish new messages to a topic that AWS IoT Analytics is subscribed to
- Log in to an Amazon Elastic Compute Cloud (Amazon EC2) instance and run commands to configure the environment
- Import the simulation data
- Run a Python script that will publish the MQTT messages to the message broker

Task 4.1 Create the AWS IoT Core rule

In this task, you use the AWS IoT Core console to create the rule for publishing IoT messages to AWS IoT Analytics. You can integrate AWS IoT Analytics with AWS IoT Core by creating a rule in AWS IoT Core. AWS IoT Core rules determine how to route data that is sent to your AWS IoT Core instance. For example, you can send IoT data to a database, such as Amazon DynamoDB. For this exercise, you will send data to AWS IoT Analytics so that you can run queries on the data.

To create the AWS IoT Core rule:

33. On the AWS Management Console, on the **Services** menu, choose **Services**.
34. From the list of services, choose **IoT Core**.

35. Choose **Get started**.

36. From the navigation menu, choose **Act**.

37. On the **Rules** menu choose **Create a rule**.

38. In the **Name** box, enter Send_IOT.

39. Copy the following SQL statement –

SELECT * FROM 'iot/aus_weather'

40. Paste the statement into the **Rule query statement** box.
    This rule will send the simulated data that you will load later in the exercise to the *channel* that you created in Task 1.

41. In the **Set one or more actions** section, choose **Add action**.

42. Select **Send a message to IoT Analytics**.

43. Choose **Configure action**.

44. Select **Manually select IoT Analytics Channel and role**.

45. To pick the channel name to send the data to, choose **Select**.

46. To choose the channel that you created earlier (*mychannel*), choose **Select**.

47. In the **Role** field, choose **Select**.

48. Choose the *IoTLabAccessRole* role.

49. Choose **Add action**
    Note: Make sure the SQL statement is still present. If it is not there, re-enter it.

50. Choose **Create rule**.
    Note: You might need to scroll down to see the **Create rule** button.

Task 4.2 Configure your environment to run the Python script

In this task, you will run a Python script to load data into the *iot/aus_weather* MQTT topic. You will connect to an existing Amazon Linux EC2 instance that already has the AWS Command Line Interface (AWS CLI) installed on it. The EC2 instance is included as part of the lab infrastructure.

Windows users should follow Task 4.2.1. Both macOS and Linux users should follow Task 4.2.2.

Task 4.2.1: Windows users – Use SSH to connect

These instructions are for Windows users only.

If you are using macOS or Linux, skip to the next section.

51. Read through the following list items in this step before you start these actions. You will not be able see these instructions when the **Details** panel is open.
    - Above these instructions you are currently reading, choose the Details dropdown menu, and then choose Show. A **Credentials** window will open.
    - Choose the **Download PPK** button and save the **labsuser.ppk** file. Typically, your browser will save it to the *Downloads* directory.
    - Exit the *Details* panel by choosing the **X**.
52. Download the required software.
    Note: You will use **PuTTY** to use Secure Shell (SSH) to access Amazon EC2 instances. If you do not have PuTTY installed on your computer, download it here.
53. Open **putty.exe**
54. Configure PuTTY to not timeout:
    - Choose **Connection**.
    - Set **Seconds between keepalives** to 30.
55. This enables you to keep the PuTTY session open for a longer period of time.
56. Configure your PuTTY session:
    - Choose **Session**.
    - **Host Name (or IP address):** For the **Command Host** instance, copy and paste the **IPv4 Public IP address**. To find it, go to the Amazon EC2 console and choose **Instances**. Select the **Command Host** instance, and in the **Description** tab, copy the **IPv4 Public IP** value.
    - Back in PuTTy, in the **Connection** list, expand **SSH**.
    - Choose **Auth** (but don't expand it).
    - Choose **Browse**.
    - Browse to and select the lab#.ppk file that you downloaded.
    - Choose **Open** to select it.
    - Choose **Open**.

57. To trust the host and connect to it, choose **Yes**.

58. When prompted **login as**, enter: ec2-user
    This will connect you to the EC2 instance.

59. Windows users: Skip ahead to the next task.

Task 4.2.2: macOS and Linux users

These instructions are for macOS and Linux users only. If you are a Windows user, skip ahead to the next task.

59. Read through the following list items in this step before you start these actions. You will not be able see these instructions when the **Details** panel is open.

    ○ Above these instructions you are currently reading, choose the Details dropdown menu, and then choose Show. A **Credentials** window will open.

    ○ Choose the **Download PEM** button and save the **labsuser.pem** file.

    ○ Exit the **Details** panel by choosing the **X**.

60. Open a terminal window, and change directory (by using cd) to the directory where the labsuser.pem file was downloaded.
    For example, if the file was saved to your *Downloads* directory, run this command:

```
cd ~/Downloads
```

61. Change the permissions on the key to be read-only by running this command:

```
chmod 400 labsuser.pem
```

62. Return to the AWS Management Console, and in the Amazon EC2 service, choose **Instances** and then select the **Command Host** instance.

63. In the **Description** tab, copy the **IPv4 Public IP** value.

64. Return to the terminal window and run this command (replace <*public-ip*> with the actual public IP address that you copied):

```
ssh -i labsuser.pem ec2-user@<public-ip>
```

65. When you are prompted to allow a first connection to this remote SSH server, enter yes.
    Because you are using a key pair for authentication, you will not be prompted for a password.

Task 4.3: Configure the Amazon EC2 environment

You must configure the EC2 instance so that you can run commands to load the IoT simulation data into the IoT pipeline. The first step is to establish the correct permissions for running the commands that will interact with AWS IoT Analytics.

Task 4.3.1: Set the AWS CLI credentials

66. Above these instructions, choose the Details dropdown menu, and then choose Show. Copy the text in the **AWS CLI** section and store it in a text editor.

67. Copy the following command and paste it in the terminal window:

```
vim ~/.aws/credentials
```

The vim editor opens. Vim is a standard text editor for Linux. If you are not familiar with it, you can get a summary of common commands at VIM Cheat sheet.

68. In the vim editor, paste the text that you copied from the AWS CLI section of the lab details page.

69. Press ESC.

70. To save the file and exit vim, enter :wq.

Task 4.3.2: Install the AWS SDK for Python (boto)

You must also configure the environment for the Python script that you will use to upload the data. The script uses the AWS software development kit (SDK) for Python (boto).

To add the SDK for Python to the environment:

71. Copy the following command –

```
pip install boto3 --user
```

72. In the terminal window, paste the command and press ENTER.

Task 4.3.3 : Download the dataset

You must also download the dataset that you will use to simulate the IoT devices.

To download the dataset:

73. Copy the following command –

```
curl -XPORT 'https://data.melbourne.vic.gov.au/resource/277b-wacc.json' > input_aus.json
```

74. In the terminal window, paste the command and press ENTER.

75. To confirm that the new input_aus.json file was created, run the ls command. You should see *input_aus.json*.

Task 4.4: Create the Python script

In this task, you run a Python script to simulate ingesting data from IoT devices.

76. Copy the following script and save it in a text editor.

```
import json

import boto3

import fileinput

import multiprocessing as mp

import os



processes = 4



# An array of boto3 IoT clients

IotBoto3Client = [boto3.client('iot-data') for i in range(processes)]



def publish_wrapper(lineID, line):

  # Select the appropriate boto3 client based on lineID

  client = IotBoto3Client[lineID % 4]



  line_read = line.strip()

  print "Publish: ", os.getpid(), lineID, line_read[:70], "..."
```

```
   payload = json.loads(line_read)



   # Publish JSON data to AWS IoT
   client.publish(
       topic='iot/aus_weather',
       qos=1,
       payload=json.dumps(payload))



if __name__ == '__main__':
   pool = mp.Pool(processes)
   jobs = []
   print "Begin Data Ingestion"
   for ID, line in enumerate(fileinput.input()):
     # Create job for each JSON object
     res = jobs.append(pool.apply_async(publish_wrapper, (ID, line)))



   for job in jobs:
     job.get()



   print "Data Ingested Successfully"
```

The script creates four SDK for Python clients, and then reads the content from the specified JavaScript Object Notation (JSON) file. The data is then published to an AWS IoT Core topic named *iot/aus_weather*. You will use this topic later in the lab. You can also download the above script here

77. Copy the following command and paste it into the command line:

vim upload_raw_data_iot.py

78. In the vim editor, paste the code that you copied at the beginning of this task.
79. To leave insert mode, press ESC.
80. To save the file and exit vim, enter :wq.

Task 5: Create a dataset

You retrieve data from a data store by creating a dataset. AWS IoT Analytics enables you to create an SQL dataset or a container dataset. In this task, you look at a simple SQL dataset that is similar to a materialized view from an SQL database. You create an SQL dataset by applying an SQL action.

To create the dataset:

81. On the AWS Management Console, on the **Services** menu, choose **Services**.
82. From the list of services, choose **IoT Analytics**.
83. On the left navigation pane, choose **Data sets**.
84. On the **Data sets** page, choose **Create a data set**.
85. Choose **Create SQL**.
86. In the **ID** field, enter my_dataset.
87. On the **Select data store source** section, choose **Edit**.
88. Choose the *my_datastore* data store that you created earlier.
89. Choose **Next**.

On the **Author SQL Query** page you see the default SQL of Select * From my_datastore.

90. On the **Author SQL Query** page, choose **Next**.
91. On the **Configure data selection filter** page, choose **Next**.
92. On the **Set query schedule** page, choose **Next**.
93. On the **Configure the results of your analytics** page, choose **Next**.

94. On the **Configure the delivery rules of your analytics results** page, choose **Create data set**.

Task 5.1: Load the data to the MQTT topic

You can add the simulation data to the MQTT topic by running the Python script that you downloaded in Task 4.3.3.

To load the data:

95. Return to the terminal window.

96. Copy the following script –

```
cat input_aus.json | jq -c '.[]' | python upload_raw_data_iot.py
```

If you encounter a Python based IndentationError, then replace the *upload_raw_data_iot.py* file with this copy.

97. In the terminal window, paste the command and press ENTER.
    You will see the data being loaded into AWS IoT Core. After the script finishes, you will see the following message: *Data Ingested Successfully*.

Task 6: Access the query results in your dataset

The dataset content is the result of your query in a file, which is in a comma-separated values (CSV) format. Before running the query to create the dataset, make sure that all messages have finished publishing to the MQTT topic.

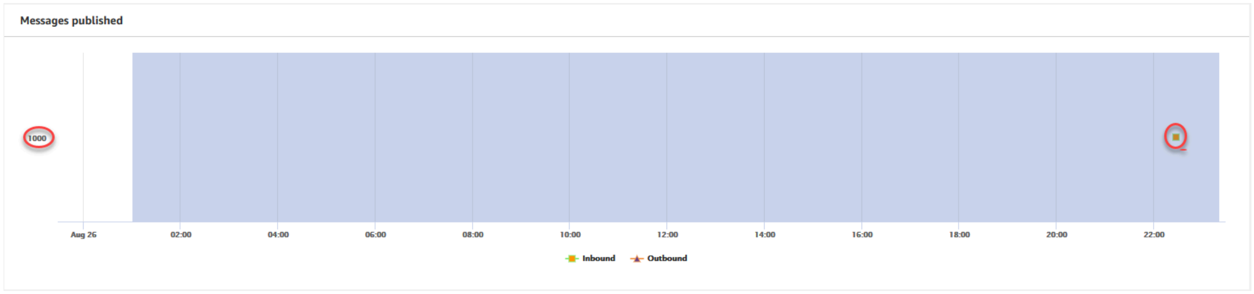
Task 6.1: Check that the messages have finished publishing

To check that the messages are all published:

98. On the AWS Management Console, on the **Services** menu, choose **Services**.

99. From the list of services, choose **IoT Core**.
    You see a page showing graphs depicting activity in your IoT environment.

100. Scroll down to see the **Messages Published** graph. You should see a graph that is similar to the following example.



Task 6.2: Access the query results

To access the query results:

101. On the AWS Management Console, on the **Services** menu, choose **Services**.

102. From the list of services, choose **IoT Analytics**.

103. In the **AWS IoT Analytics console**, in the left navigation pane, choose **Data sets**.

104. On the **Data sets** page, choose the name of the dataset that you created previously (*my_dataset*).

105. On the dataset information page, in the upper-right corner, choose **Actions** and then choose **Run now**.

106. To check if the dataset is ready, under the name of the dataset in the upper-left corner, look for *SUCCEEDED*. The **Details** section contains the query result.

107. In the left navigation pane, choose **Content**.

108. Choose **Download**.
You can either view or download the query results.

109. Choose the back arrow button to return to the **IoT Analytics Dashboard**.

Task 7: Query the dataset

After you have data in your data store, you can analyze the data by changing the SQL that you use to create datasets. In this task, you will create a dataset that finds the average temperature for the readings in your data store.

Task 7.1 Create a dataset to find the average maximum temperature

110. In the **AWS IoT Analytics console**, in the left navigation pane, choose **Data sets**.

111. Choose **Create**.

112. Choose **Create SQL**.

113. In the **ID** box, enter Findaverage.

114. In **Select data store source**, choose **Edit**.

115. Choose **my_datastore**.

116. Choose **Next**.

117. Copy the following SQL command:

```
SELECT AVG (CAST(temp_max as double)) FROM my_datastore
```

118. In the **Author SQL query** window, paste the SQL command and choose **Next**.

119. On the **Configure data selection filter** window, choose **Next**.

120. On the **Set query schedule (optional)** window, choose **Next**.

121. On the **Configure the results of your analytics** window, choose **Next** .

122. On the **Configure the delivery rules of your analytics results** window, choose **Create data set** .

123. On the **Data sets** page, choose the **Findaverage** dataset.

124. From the **Actions** menu, choose **Run now**.
    The status is under the name of the dataset, and it will change to *Creating*. After the dataset is finished, the status will change to *SUCCEEDED*.

125. To view the results from creating the dataset, choose **Content**.
    You will see the **Result preview**, which is the average maximum temperature in the dataset. The value will be *8.429*.

Challenge question

Now that you know how to use AWS IoT Analytics to query IoT data, you can practice creating some additional queries.

For this challenge, create a dataset to find the average temperature for a specific location