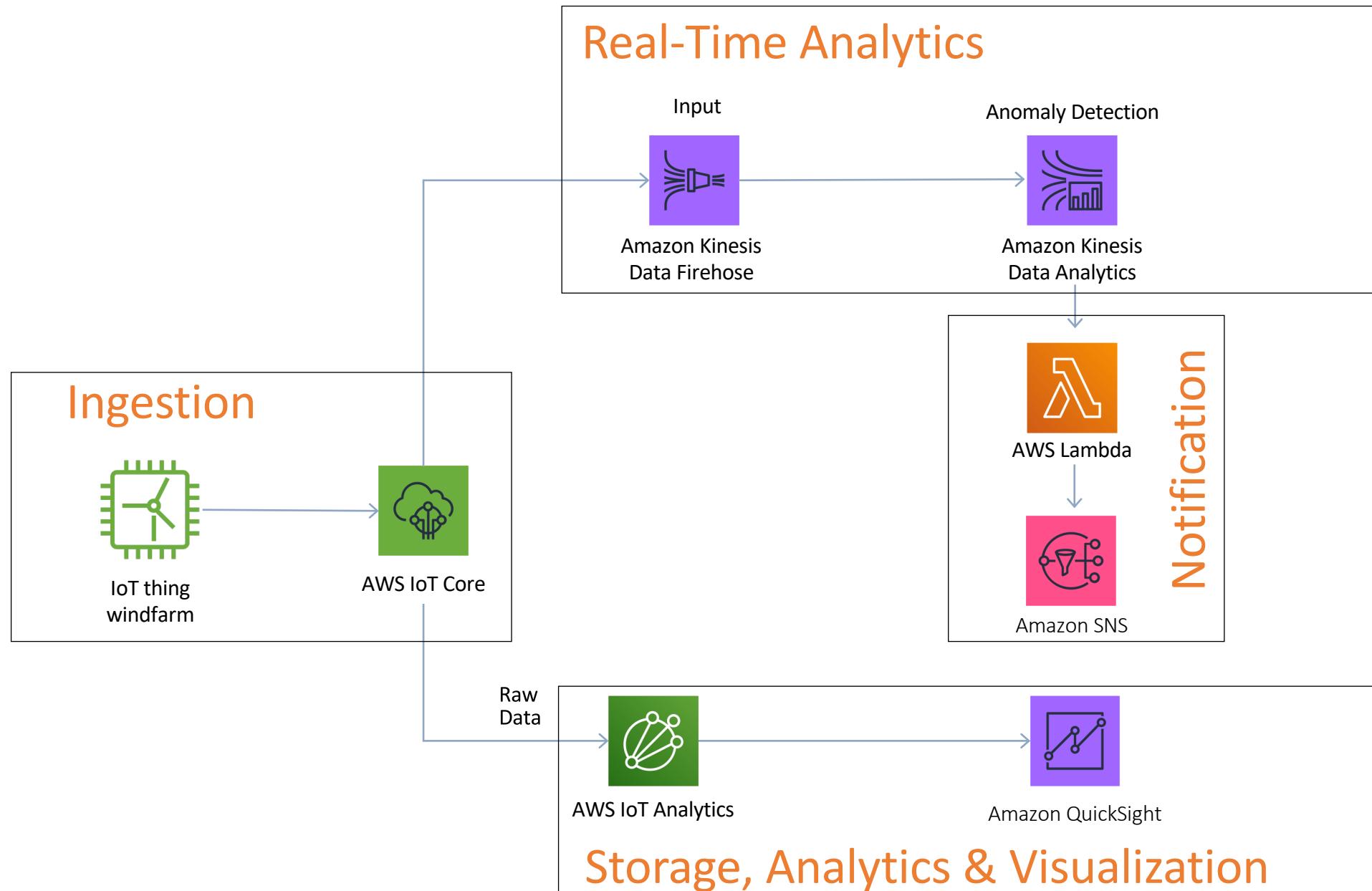


Building Predictive Applications with AWS IoT and Machine Learning

Workshop Guide

What you will build today...



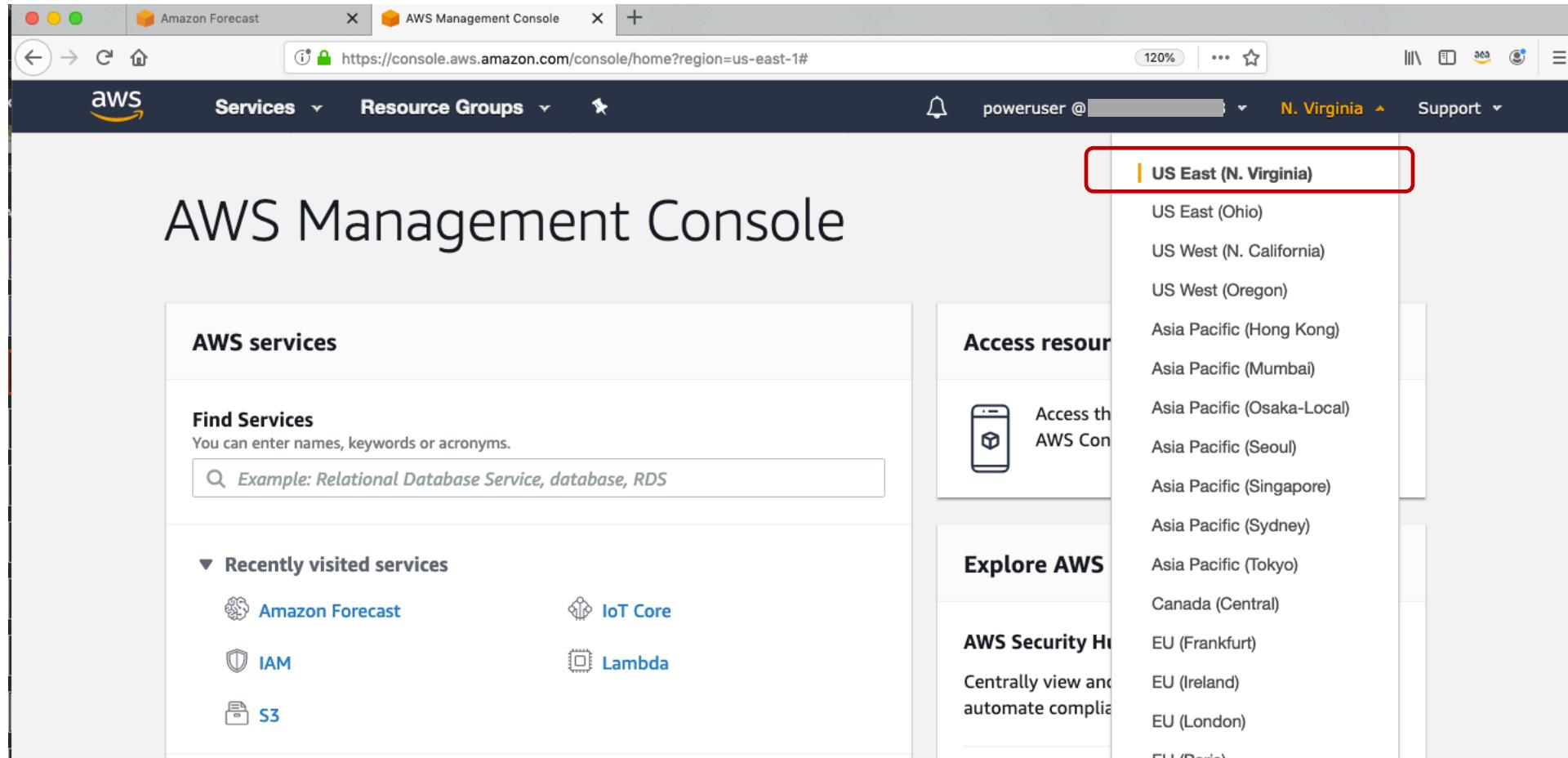
We will go over these labs:

Lab 1 –IoT ‘thing’ & Device Simulation	15 mins
Lab 2 –Kinesis Firehose and Analytics Pipeline	30 mins
Lab 3 –IoT Analytics	30 mins
Lab 4 –Notification (Optional)	30 mins

We will go over these labs:

Lab 1 –IoT ‘thing’ & Device Simulation	15 mins
Lab 2 –Kinesis Firehose and Analytics Pipeline	30 mins
Lab 3 –IoT Analytics	30 mins
Lab 4 –Notification (Optional)	30 mins

IMPORTANT - before starting... ensure that you have selected ‘N. Virginia’ as your region. The code sample provided is configured with IoT endpoint for N. Virginia, if you run this in a different region, you may have to tweak this in the code.



Setup IoT 'Thing

Select 'IoT Core' Service from AWS Console.

AWS IoT

AWS IoT is a managed cloud platform that lets connected devices - cars, light bulbs, sensor grids, and more - easily and securely interact with cloud applications and other devices.

[Get started](#)

Click Get started



Connect and manage your devices

Connect devices to the cloud using the protocol that best fits your requirements - HTTP, MQTT, or WebSocket. Devices can communicate with each other even if they are using different protocols.

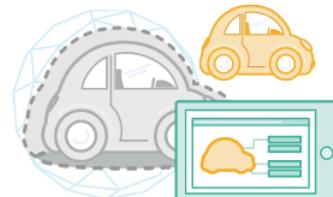
[Learn more](#)



Process and act upon device data

Filter, transform, and act upon data from devices on the fly, based on business rules. AWS IoT can be easily integrated with AWS services like Amazon DynamoDB, Amazon Kinesis, Amazon Machine Learning, and AWS Lambda.

[Learn more](#)

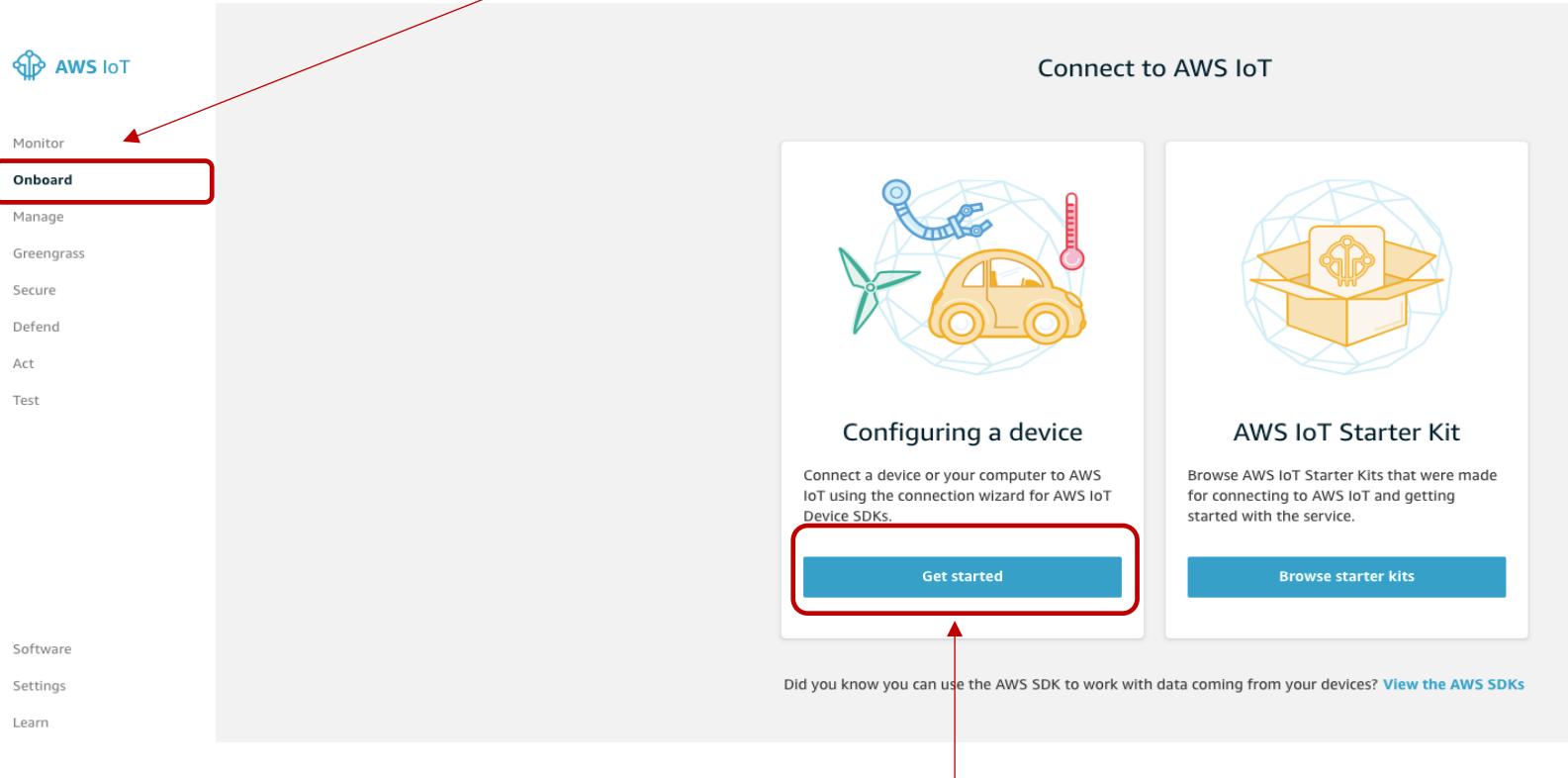


Read and set device state at any time

AWS IoT stores the latest state of a device so that it can be read or set anytime, even when the device is offline.

[Learn more](#)

On the IoT landing page, select ‘On Board’ from left navigation panel, then click ‘Get Started’ from ‘Configuring a device’ panel.



Next, click on ‘Get started’

Select 'Get Started' on next page...

The screenshot shows a guide titled "Connect to AWS IoT". It starts with a brief introduction: "Connecting a device (like a development kit or your computer) to AWS IoT requires the completion of the following steps. In this process you will:". Below this, three steps are listed, each with an icon and a numbered callout:

- 1 Register a device**
A thing is the representation and record of your physical device in the cloud. Any physical device needs a thing record in order to work with AWS IoT.
- 2 Download a connection kit**
The connection kit includes some important components: security credentials, the SDK of your choice, and a sample project.
- 3 Configure and test your device**
Using the connection kit, you will configure your device by transferring files and running a script, and test that it is connected to AWS IoT correctly.

At the bottom left, there's a link: "Want to learn more about the components of AWS IoT? Try the interactive overview". On the right side, there's a prominent blue button with white text: "Get started".

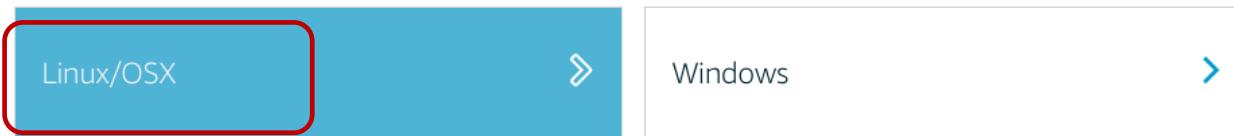


On next screen, click on 'Get started'

Select 'Linux/OSX' from platform and 'Python' from Device SDK, then click 'Next'

Select the platform and SDK that best suits how you are connecting to AWS IoT.

Choose a platform



Choose a AWS IoT Device SDK



Some prerequisites to consider:

the device should have **Python** and **Git** installed and a **TCP connection to the public internet on port 8883**.

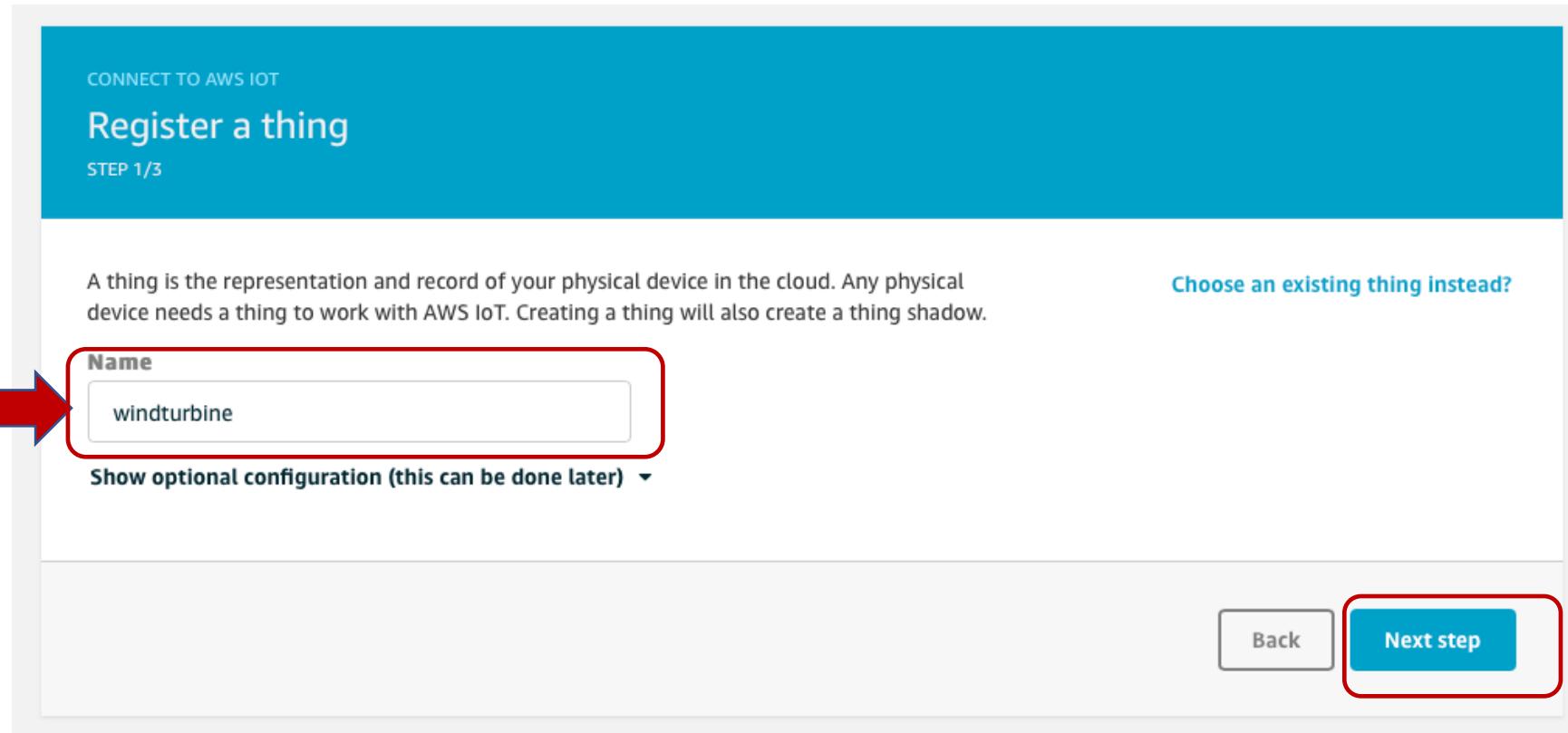
Looking for AWS IoT Device SDKs and documentation?

[View AWS IoT Device SDKs](#)

Next

Give your device a name – ‘windturbine’ then press ‘Next Step’.

Important – you should call your device windturbine (all lowercase, one word) as we refer to the device by this name in the code. Otherwise, you will need to change this in the code.



CONNECT TO AWS IOT

Download a connection kit

STEP 2/3

The following AWS IoT resources will be created:

A thing in the AWS IoT registry	windturbine	
A policy to send and receive messages	windturbine-Policy	Preview policy

The connection kit contains:

A certificate and private key	windturbine.cert.pem, windturbine.private.key
AWS IoT Device SDK	Python SDK
A script to send and receive messages	start.sh

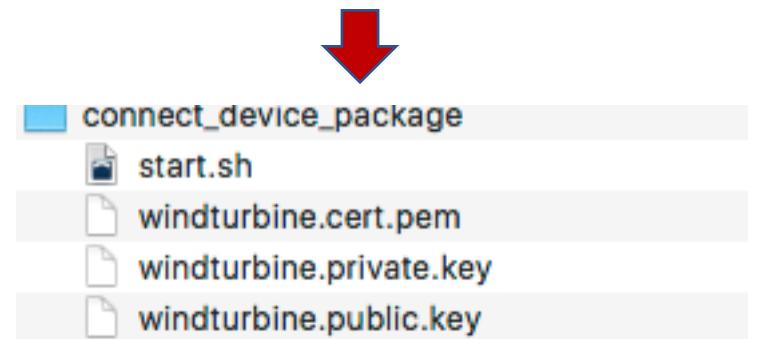
Before your device can connect and publish messages, you will need to download the connection kit.

Download connection kit for

[Linux/OSX](#) 

[Back](#)  [Next step](#)

You will see these files in the connection kit



Click here to download the connection kit

Click 'Next Step'

The next screen will show you the steps to follow, to complete the device setup

Important: Don't run these steps on your laptop. Instead, we will run these steps on Cloud9, which we will use for device simulation. Don't click the 'Done' yet.

We will setup Cloud9 in the next step.

You can leave the IoT console open as we will revisit these steps, after setting up Cloud9 IDE. You can open Cloud9 in a new browser tab or window.

The screenshot shows a step-by-step guide for connecting a device to AWS IoT. The top header reads "CONNECT TO AWS IOT" and "Configure and test your device STEP 3/3". Below the header, instructions state: "To configure and test the device, perform the following steps." Three numbered steps are listed with corresponding command-line inputs:

- Step 1: Unzip the connection kit on the device**
unzip connect_device_package.zip
- Step 2: Add execution permissions**
chmod +x start.sh
- Step 3: Run the start script. Messages from your thing will appear below**
.start.sh

At the bottom right, there are "Back" and "Done" buttons.

Setup Cloud9

Select 'Cloud9' from AWS Console. Then select 'Create environment'.

The screenshot shows the AWS Cloud9 landing page. At the top left, there's a navigation menu icon and the text "Developer Tools". Below that is the main title "AWS Cloud9" followed by the subtitle "a cloud IDE for writing, running, and debugging code". A descriptive paragraph explains that AWS Cloud9 allows you to write, run, and debug your code with just a browser, providing immediate access to a rich code editor, integrated debugger, and built-in terminal with preconfigured AWS CLI. Below this, there's a section titled "How it works" with two paragraphs. The first paragraph describes creating an AWS Cloud9 development environment on a new Amazon EC2 instance or connecting it to your own Linux server through SSH. The second paragraph explains that using the AWS Cloud9 dashboard, you can create and switch between many different AWS Cloud9 environments, each containing custom tools, runtimes, and files for a specific project. To the right of the main content area, there's a sidebar titled "Getting started" with several links: "Before you start" (2 min read), "Create a environment" (3 min read), "Working with environments" (15 min read), "Working with the IDE" (10 min read), and "Working with AWS Lambda" (5 min read). A prominent orange button labeled "Create environment" is located in the main content area, with a red arrow pointing towards it from the top right corner of the slide.

Developer Tools

AWS Cloud9

a cloud IDE for writing, running, and debugging code

AWS Cloud9 allows you to write, run, and debug your code with just a browser. With AWS Cloud9, you have immediate access to a rich code editor, integrated debugger, and built-in terminal with preconfigured AWS CLI. You can get started in minutes and no longer have to spend the time to install local applications or configure your development machine.

How it works

Create an AWS Cloud9 development environment on a new Amazon EC2 instance or connect it to your own Linux server through SSH. Once you've created an AWS Cloud9 environment, you will have immediate access to a rich code editor, integrated debugger, and built-in terminal with pre-configured AWS CLI – all within your browser.

Using the AWS Cloud9 dashboard, you can create and switch between many different AWS Cloud9 environments, each one containing the custom tools, runtimes, and files for a specific project.

New AWS Cloud9 environment

Create environment

Getting started

- Before you start 2 min read
- Create a environment 3 min read
- Working with environments 15 min read
- Working with the IDE 10 min read
- Working with AWS Lambda 5 min read



Step 1
Name environment

Step 2
Configure settings

Step 3
Review

Name environment

Environment name and description

Name
The name needs to be unique per user. You can update it at any time in your environment settings.

windturbine_simulator

Limit: 60 characters

Description - *Optional*

This will appear on your environment's card in your dashboard. You can update it at any time in your environment settings.

Write a short description for your environment

Limit: 200 characters

Cancel

Next step

Give a name to this environment

Click Next Step

AWS Cloud9

AWS Cloud9 > Environments > Create environment

Step 1 Name environment

Step 2 Configure settings

Step 3 Review

Configure settings

Environment settings

Environment type [Info](#)
Choose between creating a new EC2 instance for your new environment or connecting directly to your server over SSH.

Create a new instance for environment (EC2)
Launch a new instance in this region to run your new environment.

Connect and run in remote server (SSH)
Display instructions to connect remotely over SSH and run your new environment.

Instance type

t2.micro (1 GiB RAM + 1 vCPU)
Free-tier eligible. Ideal for educational users and exploration.

t2.small (2 GiB RAM + 1 vCPU)
Recommended for small-sized web projects.

m4.large (8 GiB RAM + 2 vCPU)
Recommended for production and general-purpose development.

Other instance type
Select an instance type.
 ▾

Platform

Amazon Linux

Ubuntu Server 18.04 LTS

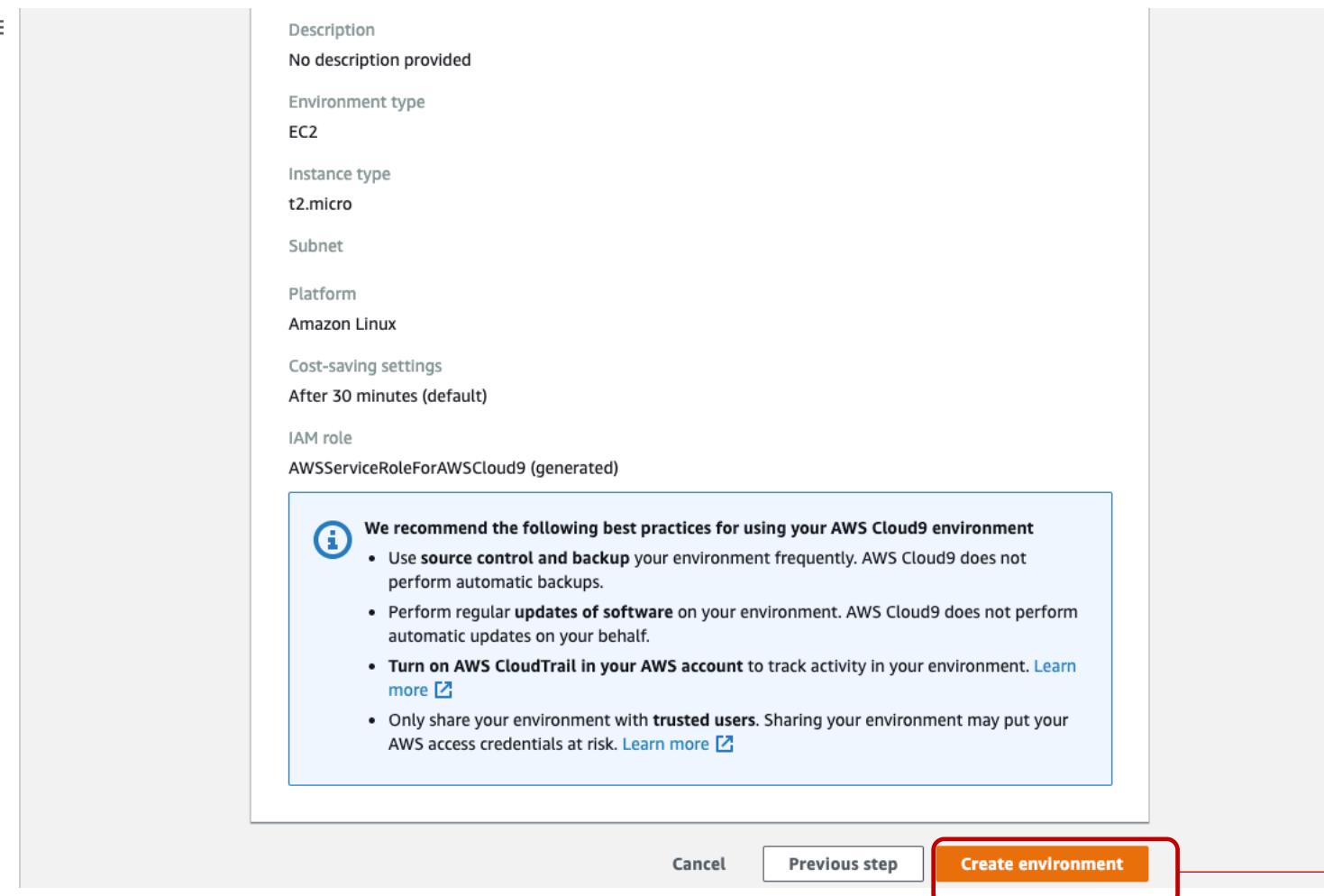
Cost-saving setting
Choose a predetermined amount of time to auto-hibernate your environment and prevent unnecessary charges. We recommend a hibernation setting of half an hour of no activity to maximize savings.

▾

IAM role

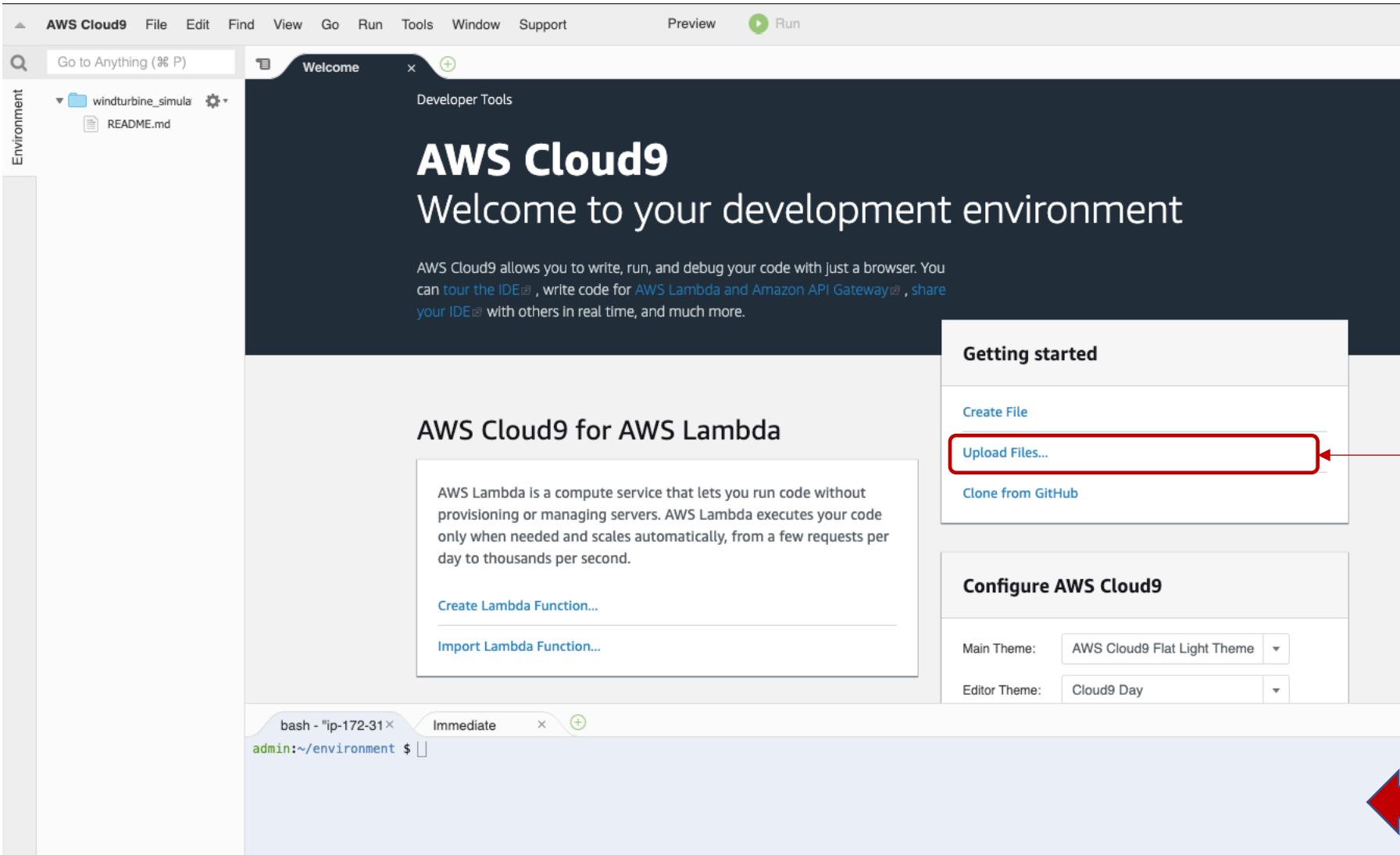
AWS Cloud9 creates a service-linked role for you. This allows AWS Cloud9 to call other AWS services on your behalf. You can change this role at any time.

Leave all settings as default and click **Next**.



Click 'Create environment' on the next screen.

It will take a few minutes before Cloud9 environment is ready...



We will upload the files that we downloaded from IoT Core here

← This is terminal window

The screenshot shows the AWS Cloud9 interface with a terminal window titled 'bash - "ip-172-31"' running on a Linux environment. The terminal displays the following command sequence:

```
admin:~/environment $ pwd  
/home/ec2-user/environment  
admin:~/environment $ ls -l  
total 8  
-rw-r--r-- 1 ec2-user ec2-user 3552 Sep 16 19:44 connect_device_package.zip  
-rw-r--r-- 1 ec2-user ec2-user 569 Sep 3 11:04 README.md  
admin:~/environment $ unzip connect_device_package.zip  
Archive: connect_device_package.zip  
  inflating: windturbine.private.key  
  inflating: windturbine.public.key  
  inflating: windturbine.cert.pem  
  inflating: start.sh  
admin:~/environment $ ls -l  
total 24  
-rw-r--r-- 1 ec2-user ec2-user 3552 Sep 16 19:44 connect_device_package.zip  
-rw-r--r-- 1 ec2-user ec2-user 569 Sep 3 11:04 README.md  
-rw-rw-r-- 1 ec2-user ec2-user 847 Sep 16 19:23 start.sh  
-rw-rw-r-- 1 ec2-user ec2-user 1220 Sep 16 19:23 windturbine.cert.pem  
-rw-rw-r-- 1 ec2-user ec2-user 1675 Sep 16 19:23 windturbine.private.key  
-rw-rw-r-- 1 ec2-user ec2-user 451 Sep 16 19:23 windturbine.public.key  
admin:~/environment $
```

Three specific lines of the terminal output are highlighted with red boxes and arrows pointing to explanatory text on the right:

- The line 'connect_device_package.zip' is highlighted, with an arrow pointing to the text: 'First, we confirm the package (connect_device_package) is uploaded to cloud9.'
- The line 'unzip connect_device_package.zip' is highlighted, with an arrow pointing to the text: 'Then we unzip the file'
- The line 'Archive: connect_device_package.zip' is highlighted, with an arrow pointing to the text: 'The content in this package includes the public and private keys, device certificate and 'start.sh' script that will setup the device and get it ready for use.'

First, we confirm the package (connect_device_package) is uploaded to cloud9.

Then we unzip the file

The content in this package includes the public and private keys, device certificate and 'start.sh' script that will setup the device and get it ready for use.

Configure and test your device

STEP 3/3

To configure and test the device, perform the following steps.

Step 1: Unzip the connection kit on the device`unzip connect_device_package.zip`**Step 2: Add execution permissions**`chmod +x start.sh`**Step 3: Run the start script. Messages from your thing will appear below**`./start.sh`

Waiting for messages from your device



Now go back to the IoT Console, copy step 2 and 3, and run on Cloud9 terminal to finish setting up the IoT device.

Click 'Done', after running this.

Important: ensure to run this script using sudo
`sudo ./start.sh`

```

AWS Cloud9 File Edit Find View Go Run Tools Window Support Preview Run

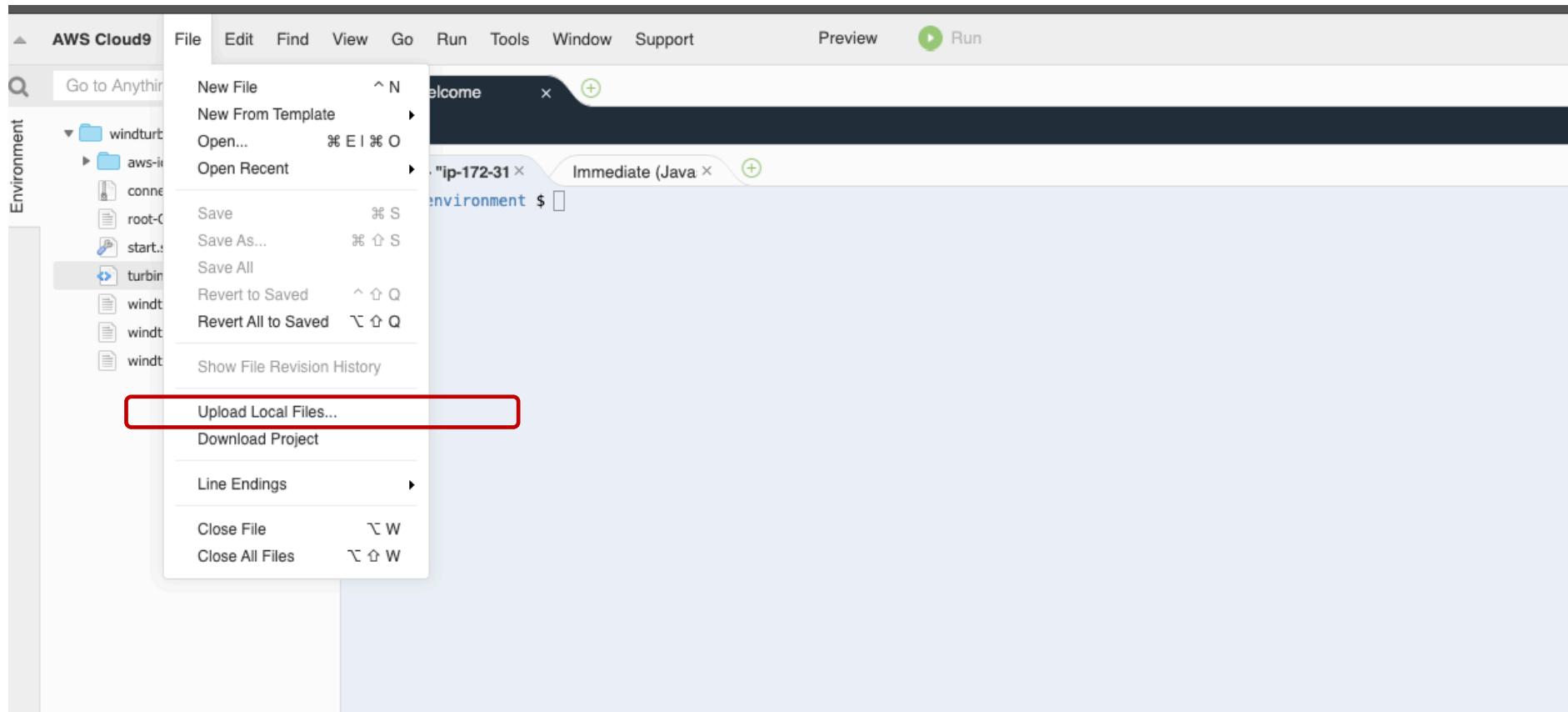
Go to Anything (% P)
Welcome Developer Tools
AWS Cloud9
Welcome to your development environment

bash - "ip-172-31" Immediate
Admin:~/environment $ unzip connect_device_package.zip
Archive: connect_device_package.zip
  inflating: windturbine.private.key
  inflating: windturbine.public.key
  inflating: windturbine.cert.pem
  inflating: start...
Admin:~/environment $ chmod +x start.sh
Admin:~/environment $ sudo ./start.sh

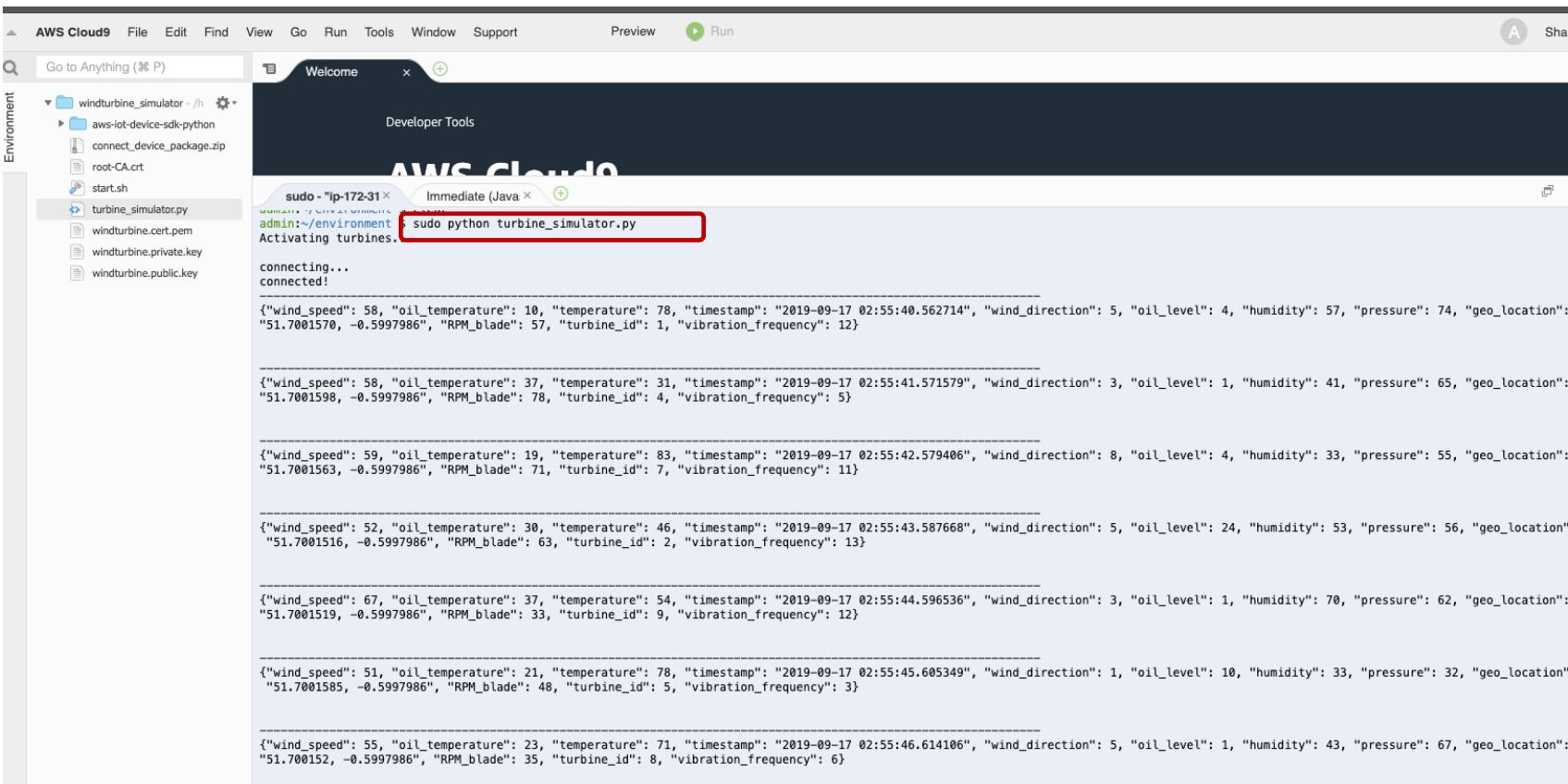
```

When you run script `start.sh`, it downloads the AWS IoT SDK and starts sending sample messages to AWS IoT core.

Now that we have setup the device on Cloud9, you are ready to upload the device simulation code (*turbine_simulator.py*), that you downloaded with this guide.



Now, run the python script...
sudo python turbine_simulator.py



The screenshot shows the AWS Cloud9 IDE interface. On the left, there's a sidebar labeled "Environment" with a file tree containing files like "windturbine_simulator", "aws-iot-device-sdk-python", "connect_device_package.zip", "root-CA.crt", "start.sh", "turbine_simulator.py", "windturbine.cert.pem", "windturbine.private.key", and "windturbine.public.key". The main window has tabs for "Welcome" and "Developer Tools". A terminal window titled "sudo - ip-172-31" is active, showing the command "sudo python turbine_simulator.py" being run. The output of the script is displayed below, showing simulated device data points being sent to AWS IoT Core.

```
sudo - ip-172-31 x Immediate (Java x +)
admin:~/environment : sudo python turbine_simulator.py
Activating turbines.

connecting...
connected!

{"wind_speed": 58, "oil_temperature": 10, "temperature": 78, "timestamp": "2019-09-17 02:55:40.562714", "wind_direction": 5, "oil_level": 4, "humidity": 57, "pressure": 74, "geo_location": "51.7001570, -0.5997986", "RPM blade": 57, "turbine_id": 1, "vibration_frequency": 12}

{"wind_speed": 58, "oil_temperature": 37, "temperature": 31, "timestamp": "2019-09-17 02:55:41.571579", "wind_direction": 3, "oil_level": 1, "humidity": 41, "pressure": 65, "geo_location": "51.7001598, -0.5997986", "RPM blade": 78, "turbine_id": 4, "vibration_frequency": 5}

{"wind_speed": 59, "oil_temperature": 19, "temperature": 83, "timestamp": "2019-09-17 02:55:42.579406", "wind_direction": 8, "oil_level": 4, "humidity": 33, "pressure": 55, "geo_location": "51.7001563, -0.5997986", "RPM blade": 71, "turbine_id": 7, "vibration_frequency": 11}

{"wind_speed": 52, "oil_temperature": 30, "temperature": 46, "timestamp": "2019-09-17 02:55:43.587668", "wind_direction": 5, "oil_level": 24, "humidity": 53, "pressure": 56, "geo_location": "51.7001516, -0.5997986", "RPM blade": 63, "turbine_id": 2, "vibration_frequency": 13}

 {"wind_speed": 67, "oil_temperature": 37, "temperature": 54, "timestamp": "2019-09-17 02:55:44.596536", "wind_direction": 3, "oil_level": 1, "humidity": 70, "pressure": 62, "geo_location": "51.7001519, -0.5997986", "RPM blade": 33, "turbine_id": 9, "vibration_frequency": 12}

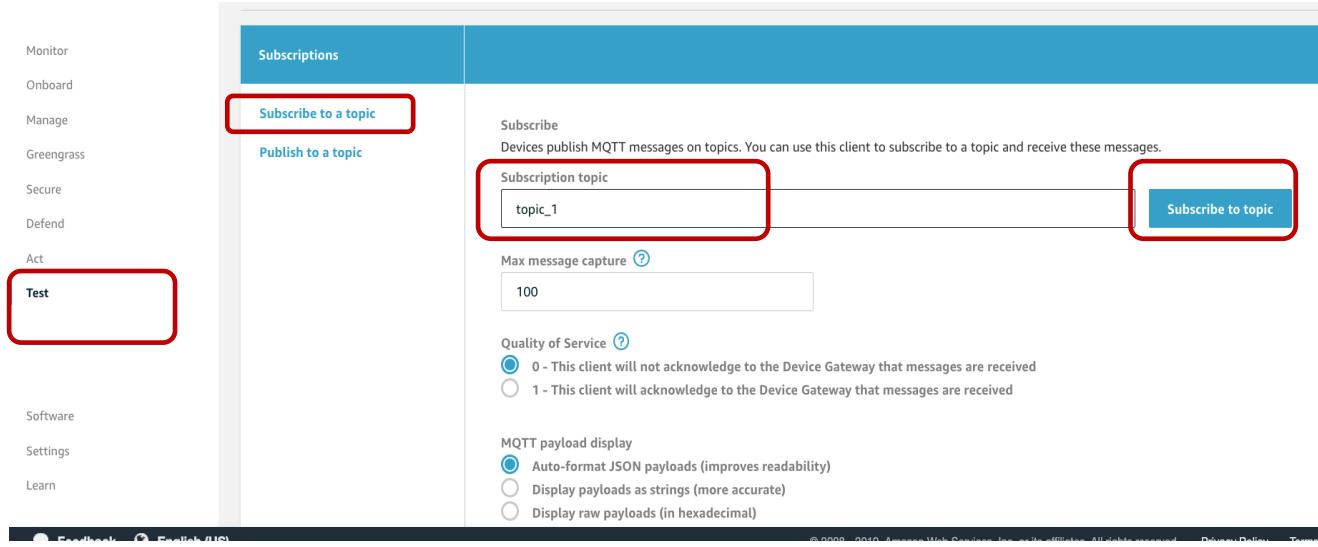
 {"wind_speed": 51, "oil_temperature": 21, "temperature": 78, "timestamp": "2019-09-17 02:55:45.605349", "wind_direction": 1, "oil_level": 10, "humidity": 33, "pressure": 32, "geo_location": "51.7001585, -0.5997986", "RPM blade": 48, "turbine_id": 5, "vibration_frequency": 3}

 {"wind_speed": 55, "oil_temperature": 23, "temperature": 71, "timestamp": "2019-09-17 02:55:46.614106", "wind_direction": 5, "oil_level": 1, "humidity": 43, "pressure": 67, "geo_location": "51.700152, -0.5997986", "RPM blade": 35, "turbine_id": 8, "vibration_frequency": 6}
```

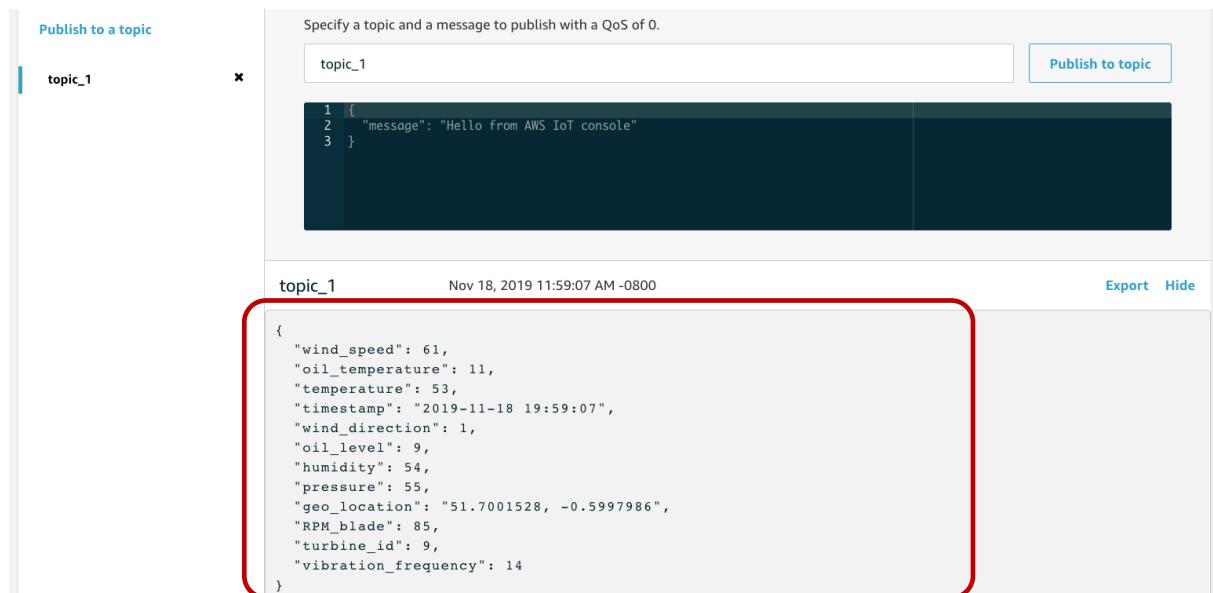
The simulated device is now sending data to AWS IoT Core on the topic “topic_1”. Please note: Let the simulator run for the entire duration of this workshop

This Python script generates telemetry data points using a random function and the anomalies are introduced after few seconds in a loop.

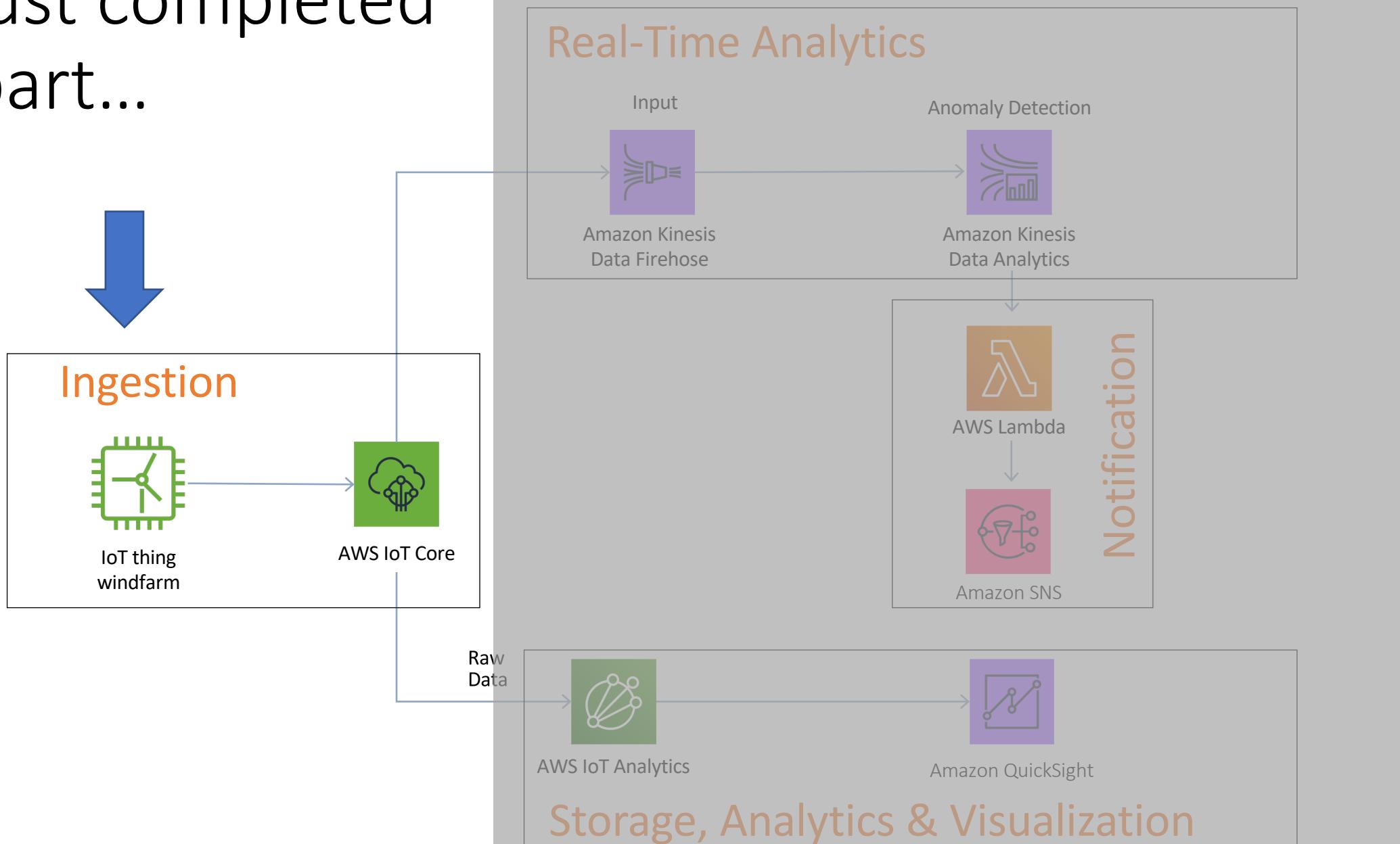
You can test the data is flowing to AWS IoT Core, by selecting 'Test' from the left hand panel and subscribing to topic_1 topic



After subscribing, you can see the incoming data in the IoT Core console, as shown below...



You just completed this part...

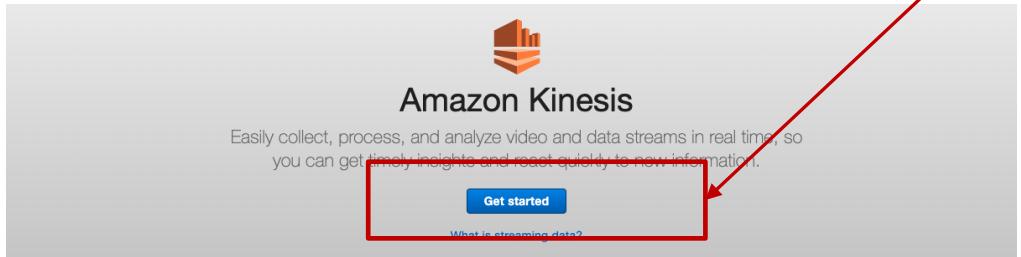


We will go over these labs:

Lab 1 –IoT ‘thing’ & Device Simulation	15 mins
Lab 2 –Kinesis Firehose and Analytics Pipeline	30 mins
Lab 3 –IoT Analytics	30 mins
Lab 4 –Notification (Optional)	30 mins

Set up Kinesis Firehose

We will first create delivery stream, that will take input from AWS IoT.
Select 'Kinesis Firehose' from the Kinesis Console, then click 'Get started'...



...on the next screen, click 'Create delivery stream'

The screenshot shows the "Get started with Amazon Kinesis" page. At the top, there's a navigation bar with the AWS logo, "Services", and "Resource Groups". The main heading is "Get started with Amazon Kinesis" with the sub-instruction "To get started, choose an Amazon Kinesis resource to create." Below this, there are four main sections: "Ingest and process streaming data with Kinesis streams", "Deliver streaming data with Kinesis Firehose delivery streams", "Analyze streaming data with Kinesis analytics applications", and "Ingest and process media streams with Kinesis video streams". Each section contains a diagram and descriptive text. A large red arrow points from the "Get started" button on the previous screen down to the "Create delivery stream" button in this screen, which is also highlighted with a red rectangular box.

Get started with Amazon Kinesis

To get started, choose an Amazon Kinesis resource to create.

Ingest and process streaming data with Kinesis streams

Process data with your own applications, or using AWS managed services like Amazon Kinesis Data Firehose, Amazon Kinesis Data Analytics, or AWS Lambda.

Deliver streaming data with Kinesis Firehose delivery streams

Continuously collect, transform, and load streaming data into destinations such as Amazon S3 and Amazon Redshift.

Create delivery stream

Analyze streaming data with Kinesis analytics applications

Run continuous analysis on streaming data from Kinesis data streams and Kinesis Firehose delivery streams.

Create analytics application

Ingest and process media streams with Kinesis video streams

Build applications to process or analyze streaming media.

Create video stream

Step1: Name and Source

Kinesis Firehose - Create delivery stream

Step 1: Name and source

Step 2: Process records

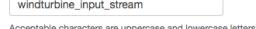
Step 3: Choose destination

Step 4: Configure settings

Step 5: Review

New delivery stream

Delivery streams load data, automatically and continuously, to the destinations that you specify. Kinesis Firehose resources are not covered under the [AWS Free Tier](#), and [usage-based charges apply](#). For more information, see [Kinesis Firehose pricing](#).

Delivery stream name* 

Acceptable characters are uppercase and lowercase letters, numbers, underscores, hyphens, and periods.

Choose source

Choose how you would prefer to send records to the delivery stream.

Firehose data flow overview

```
graph LR; Source[Source] --> Firehose[Firehose delivery stream]; Firehose --> Destination[Destination]
```

Source* Direct PUT or other sources
Choose this option to send records directly to the delivery stream, or to send records from AWS IoT, CloudWatch Logs, or CloudWatch Events.
 Kinesis stream

Give a name to this firehouse, we call it
windturbine_input_stream

Leave other options default, and click 'next'....

Step2: Process Records

Step 1: Name and source

Step 2: Process records

Step 3: Choose destination

Step 4: Configure settings

Step 5: Review

Process records

Kinesis Firehose can transform records or convert record format before delivery.

Process records data flow overview

```
graph LR; Source[Source records] --> Transform[Transform source records]; Transform --> Convert[Convert record format]; Convert --> Destination[Destination]
```

Source records
Transform source records
Invoke AWS Lambda Function
Refer to AWS Glue table for schema

Transform source records with AWS Lambda

To return records from AWS Lambda to Kinesis Firehose after transformation, the Lambda function you invoke must be compliant with the required record transformation output model. [Learn more](#)

Record transformation* Disabled
 Enabled

Convert record format

Data in Apache parquet or Apache ORC format is typically more efficient to query than JSON. Kinesis Data Firehose can convert your JSON-formatted source records using a schema from a table defined in [AWS Glue](#). For records that aren't in JSON format, create a Lambda function that converts them to JSON in the [Transform source records with AWS Lambda](#) section above. [Learn more](#)

Record format conversion* Disabled
 Enabled
If record format conversion is enabled, Firehose can deliver data to Amazon S3 only. Record format conversion will be configured using the OpenX JSON SerDe. For other options use the AWS CLI.

* Required

Cancel Previous  Next

...on the next page (step 2), leave all settings default and click 'Next' again.

Step3: Choose Destination

Kinesis Firehose - Create delivery stream

Step 1: Name and source
Step 2: Process records
Step 3: Choose destination
Step 4: Configure settings
Step 5: Review

Select destination

Destination* Amazon S3

Amazon S3 is object storage built to store and retrieve any amount of data from anywhere on the web.

Amazon Redshift

Amazon Redshift is a fast, fully managed, petabyte-scale data warehouse that makes it simple and cost effective to analyze all your data using your existing business intelligence tools

Amazon Elasticsearch Service

Elasticsearch is an open-source search and analytics engine for use cases such as log analytics, real-time application monitoring, and click stream analytics

Splunk

Splunk is an operational intelligence tool for analyzing machine-generated data in real-time

Firehose to S3 data flow overview

Source → Firehose delivery stream → S3 bucket (destination)

S3 destination

Choose a destination in Amazon S3 where your data will be stored. Amazon S3 is object storage built to store and retrieve any amount of data from anywhere. [Learn more](#)

S3 bucket*

At this step we will choose destination 'Amazon S3', to store incoming telemetry as a raw data.

You can choose an existing S3 Buckets or create a new one. We will create a new bucket, we can call it '[`<yourname123>-raw-data`](#)'. Remember to add your name or a unique prefix or suffix to this bucket name as bucket names are unique.

We can leave all other settings default and click next.

Step4: Configure Setting

S3 encryption* Disabled Enabled

Error logging
Firehose can log record delivery errors to CloudWatch Logs. If enabled, a CloudWatch log group and corresponding log streams are created on your behalf. [Learn more](#)

Error logging* Disabled Enabled

Tags (optional)
You can add tags to organize your AWS resources, track costs, and control access. [Learn more](#)

Key Value - optional
 Enter key Enter value Remove tag
 Add tag
You can add 49 more tag(s)

IAM role
Firehose uses an IAM role to access your specified resources, such as the S3 bucket and KMS key. [Learn more](#)

IAM role* Create new or choose

* Required Cancel Previous Next

On the next page (Step 4), scroll down to the bottom. If you do not have a existing IAM role for Kinesis Firehose, we will need to create one. This role will be attached to the Kinesis Firehose and enable it to integrate with other other AWS services.

Click Next

Amazon Kinesis Firehose is requesting permission to use resources in your account

Click Allow to give Amazon Kinesis Firehose Read and Write access to resources in your account.

▼ Hide Details

Role Summary [?](#)

Role Description	Provides access to AWS Services and Resources
IAM Role	<input type="button"/> Create a new IAM Role
Role Name	firehose_delivery_role

[View Policy Document](#)

When you select 'create a new role', you will see the screenshot on the left. You can change the role name or leave it default. Click 'Allow' and continue.

Step5: Review and create Firehose

Step 3: Choose destination
Step 4: Configure settings
Step 5: Review

Name and source

Delivery stream name: windturbine_input_stream

Source: Direct PUT or other sources
After creating the delivery stream, send records directly to the delivery stream, or send records from AWS IoT, CloudWatch Logs, or CloudWatch Events.

Process records

Source record transformation: Disabled
Record format conversion: Disabled

Destination

Destination: Amazon S3
S3 bucket: neeraku-raw-data
S3 bucket Prefix: no prefix specified
S3 bucket error prefix: no error prefix specified

Settings

S3 buffer conditions: 5 MB or 300 seconds
Compression: Disabled
Encryption: Disabled
Error logging: Enabled
Tags: no tags specified
IAM role: firehose_delivery_role

Cancel Previous **Create delivery stream**

Finally, review the settings (Step 5) and click 'Create delivery stream'

Amazon Kinesis

Services Resource Groups N. Virginia

Dashboard Data Streams Data Firehose Data Analytics Video Stream External resources What's new

Firehose delivery streams

Kinesis Firehose delivery streams continuously collect, transform, and load streaming data into the destinations that you specify.

Successfully created delivery stream windturbine_input_stream
Next, send records directly to the delivery stream using the Amazon Kinesis Agent or the Firehose API using the AWS SDK, or send records from AWS IoT, CloudWatch Logs, or CloudWatch Events. Learn more

Create delivery stream Test with demo data Delete

Filter Firehose delivery streams

Name	Status	Created	Source	Record transformation	Destination
windturbine_input_stream	Active	2019-09-06T14:45:0700	Direct PUT and other sources	Disabled	Amazon S3 neeraku-raw-data

You will now see it on the Firehose Dashboard.

Setup IoT Rule (Connect IoT Core to Kinesis Firehose)



Monitor

Onboard

Manage

Greengrass

Secure

Defend

Act

Test



You don't have any rules yet

Rules give your things the ability to interact with AWS and other web services. Rules are analyzed and actions are performed based on the messages sent by your things.

Learn more

Create a rule

Go back to the IoT console, and select 'Act' from the left panel.

Click 'Create a rule'

Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name

turbine_to_firehose



Give this rule a name e.g. turbine_to_firehose

Description



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 temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see [AWS IoT SQL Reference](#).

```
1 SELECT * from topic_1
```

Write SELECT * FROM 'topic_1'
In this box

Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*.required)

Add action

Important: Ensure the topic name is in single quotes.

Click Add actions

-  Send a message to an SQS queue
SQS
-  Send a message to an Amazon Kinesis Stream
AMAZON KINESIS
-  Republish a message to an AWS IoT topic
AWS IOT REPUBLISH
-  Store a message in an Amazon S3 bucket
S3
-  Send a message to an Amazon Kinesis Firehose stream
AMAZON KINESIS FIREHOSE
-  Send message data to CloudWatch
CLOUDWATCH METRICS
-  Change the state of a CloudWatch alarm
CLOUDWATCH ALARMS
-  Send a message to the Amazon Elasticsearch Service
AMAZON ELASTICSEARCH
-  Send a message to a Salesforce IoT Input Stream
SALESFORCE IOT
-  Send a message to IoT Analytics
IOT ANALYTICS
-  Send a message to an IoT Events Input
IOT EVENTS
-  Start a Step Functions state machine execution
STEP FUNCTIONS



Select Amazon Kinesis Firehose.

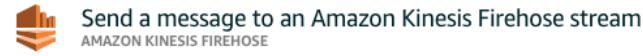
Note: Ensure you are selecting Kinesis Firehose stream and NOT Kinesis Stream

Cancel

Configure action

Click 'Configure Action'.

Configure action



This will send the message to an Amazon Kinesis Firehose stream.

*Stream name

Separator

Create a new resource

Select 'windturbine_input_stream', we created earlier.

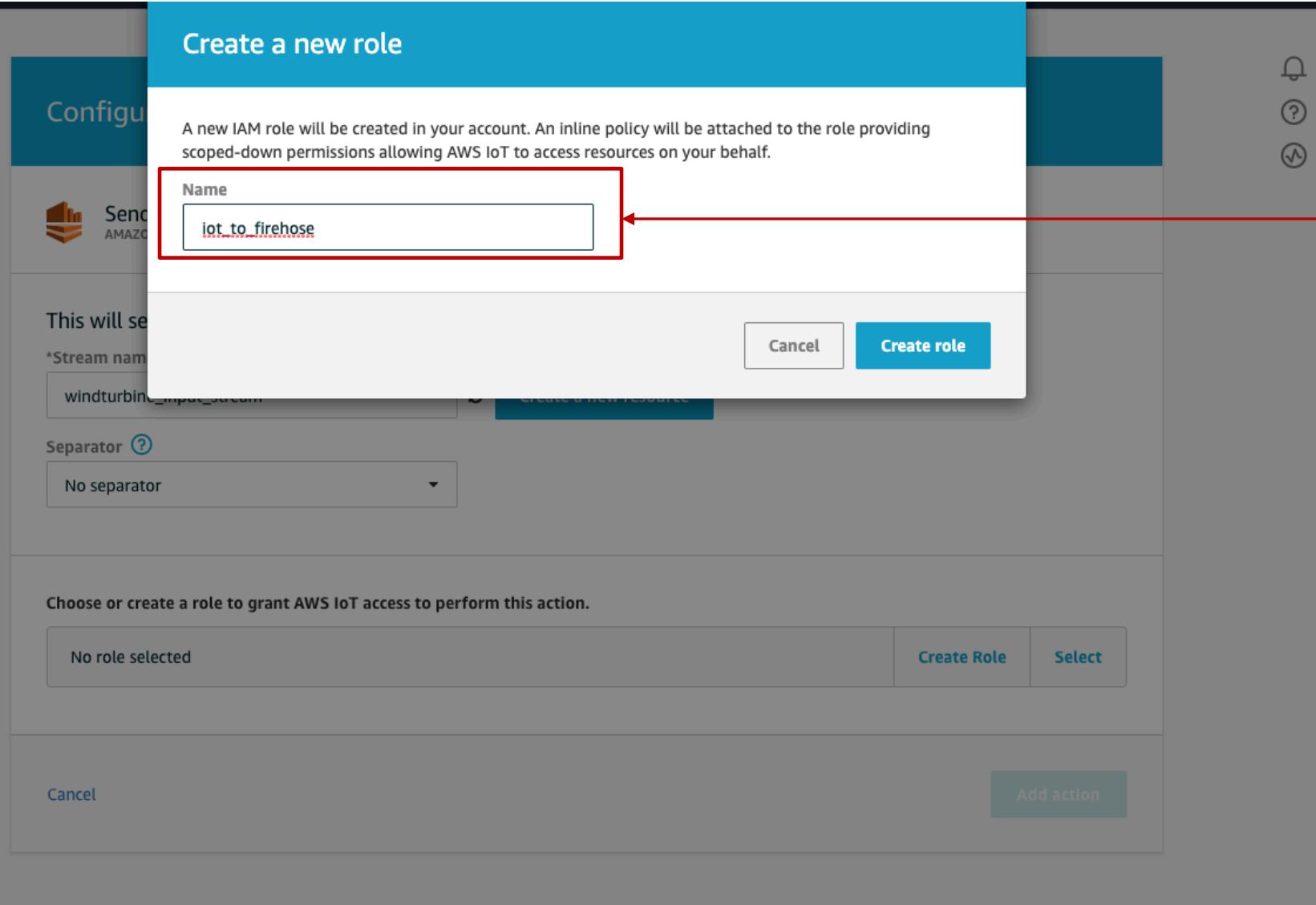
Choose or create a role to grant AWS IoT access to perform this action.

No role selected

Create Role Select

Cancel

Add action



Create an IAM role to enable IoT Core to access Kinesis Firehose

Configure action

Send a message to an Amazon Kinesis Firehose stream AMAZON KINESIS FIREHOSE

This will send the message to an Amazon Kinesis Firehose stream.

*Stream name



Create a new resource

Separator 

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

iot_to_firehose

Policy Attached ✓

Create Role

Select

[Cancel](#)

Add action

Click 'Add Action'

Rule query statement

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see [AWS IoT SQL Reference](#).

```
1 SELECT * from topic_1
```



Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*.required)



Send a message to an Amazon Kinesis Firehose stream
windturbine_input_stream

[Remove](#) [Edit](#)

[Add action](#)

Error action

Optionally set an action that will be executed when something goes wrong with processing your rule.

[Add action](#)

Tags

Apply tags to your resources to help organize and identify them. A tag consists of a case-sensitive key-value pair. [Learn more](#) about tagging your AWS resources.

Tag name

Provide a tag name, e.g. Manufacturer

Value

Provide a tag value, e.g. Acme-Corporation

[Clear](#)

[Add another](#)

[Cancel](#)

[Create rule](#)

Amazon Kinesis Firehose stream is now setup as target where IoT rules engine will push the incoming data.

Click 'Create rule' to complete this step

Setup Kinesis Data Analytics Application

On Kinesis dashboard, select 'Data Analytics'.

The screenshot shows the Amazon Kinesis Analytics dashboard. On the left, a sidebar menu lists 'Amazon Kinesis' (Dashboard, Data Streams, Data Firehose, **Data Analytics**, Video Streams), 'External resources' (What's new), and 'Analytics documentation and support' (Getting started guide, Analytics documentation, Analytics support, Forums). The main content area is titled 'Amazon Kinesis Analytics' with the sub-instruction: 'Run continuous analysis on streaming data in real-time from Amazon Kinesis Streams and Firehose.' Below this are three main sections: 'Generate time-series analytics' (Calculate performance metrics over time windows, and stream values to Amazon S3 or Amazon Redshift through an Amazon Kinesis Firehose delivery system.), 'Feed real-time dashboards' (Send aggregated and processed streaming results downstream to feed real-time dashboards.), and 'Create real-time metrics' (Create custom metrics and triggers for use in real-time monitoring, notifications, and alarms.). A prominent blue button labeled 'Create application' with a 'Getting started guide' link is centered at the bottom of the main content area. A red arrow points from the top-left text instruction to the 'Data Analytics' menu item in the sidebar. Another red arrow points from the bottom text instruction to the 'Create application' button.

Amazon Kinesis Analytics

Run continuous analysis on streaming data in real-time from Amazon Kinesis Streams and Firehose.

Create application
Getting started guide

Generate time-series analytics

Calculate performance metrics over time windows, and stream values to Amazon S3 or Amazon Redshift through an Amazon Kinesis Firehose delivery system.

Feed real-time dashboards

Send aggregated and processed streaming results downstream to feed real-time dashboards.

Create real-time metrics

Create custom metrics and triggers for use in real-time monitoring, notifications, and alarms.

Analytics documentation and support

- [Getting started guide](#)
- [Analytics documentation](#)
- [Analytics support](#)
- [Forums](#)

Click 'Create application' to get started

Amazon Kinesis

Dashboard

Data Streams

Data Firehose

Data Analytics

Video Streams

External resources

What's new

Kinesis Analytics - Create application

Kinesis Analytics applications continuously read and analyze data from a connected streaming source in real-time. To enable interactivity with your data during configuration you will be prompted to run your application. Kinesis Analytics resources are not covered under the [AWS Free Tier](#), and [usage-based charges apply](#). For more information, see [Kinesis Analytics pricing](#).

Application name*

windturbine_analytics_app

Description



Runtime

SQL

Apache Flink 1.6

* Required

Cancel

Create application

Give your app a name
e.g. windturbine_analytics_app.

Choose SQL Runtime

windturbine_analytics_app

Application ARN: arn:aws:kinesisanalytics:us-east-1:796984966265:application/windturbine_analytics_app
Application version ID: 1 ⓘ



Successfully created Application windturbine_analytics_app



Next, choose Connect streaming data.



Source

Streaming data

Connect to an existing Kinesis stream or Firehose delivery stream, or easily create and connect to a new demo Kinesis stream. Each application can connect to one streaming data source. [Learn more](#)

Connect streaming data



At this step, we will connect to the source, Kinesis Data Firehose ([windturbine_input_stream](#)) that we created in the previous step.

Reference data (optional)

Enrich data from your streaming data source with JSON or CSV data stored as an object in Amazon S3. Each application can connect to one reference data source.



Real time analytics

Author your own SQL queries or add SQL from templates to easily analyze your source data.

Important: Ensure the data simulator is on and it is sending data into Kinesis Firehose. This will enable Kinesis Data Analytics App to discover the schema.

Connect streaming data source

Choose from your Kinesis streams and Firehose delivery streams, or quickly configure a demo Kinesis stream that can be used to explore Kinesis Analytics.

[Choose source](#) [Configure a new stream](#)

Source* Kinesis stream ⓘ
 Kinesis Firehose delivery stream ⓘ

Kinesis Firehose delivery stream*  [Create new](#) 

[View windturbine_input_stream in Kinesis Firehose](#) 

In-application stream name In your SQL queries, refer to this source
as:

SOURCE_SQL_STREAM_001

Record pre-processing with AWS Lambda

Kinesis Analytics can invoke your Lambda function to pre-process records before they are used in this application. To pre-process records, your Lambda function must be compliant with the required record transformation output model. [Learn more](#)

Record pre-processing* Disabled
 Enabled

Access permissions

Create or choose IAM role with the required permissions. [Learn more](#)

Access permissions* Create / update IAM role kinesis-analytics-windturbine_analytics_app-us-east-1
 Choose from IAM roles that Kinesis Analytics can assume

Select 'windturbine_input_stream'

...scroll down

Access permissions

Create or choose IAM role with the required permissions. [Learn more](#)

Access permissions* Create / update IAM role kinesis-analytics-windturbine_analytics_app-us-east-1 Choose from IAM roles that Kinesis Analytics can assume

Schema

Schema discovery can generate a schema using recent records from the source. Schema column names are the same as in the source, unless they contain special characters, repeated column names, or reserved keywords. [Learn more](#)

[Edit schema](#) [Retry schema discovery](#)

Raw | Lambda output | Formatted (Selected)

Q. Filter by column name or column type									
wind_speed INTEGER	oil_temperature INTEGER	temperature INTEGER	COL_timestamp TIMESTAMP	wind_direction INTEGER	oil_level INTEGER	humidity INTEGER	pressure INTEGER	geo_location VARCHAR(32)	
40	13	38	2019-09-28 19:26:59.156	4	19	64	49	51.7001526, -0.	
40	34	40	2019-09-28 19:27:12.408	2	13	33	51	51.7001576, -0.	
59	20	64	2019-09-28 19:27:01.206	4	2	53	72	51.7001593, -0.	
57	22	32	2019-09-28 19:27:10.374	8	10	57	40	51.7001556, -0.	
75	25	55	2019-09-28 19:26:56.106	6	13	41	46	51.7001520, -0.	
68	43	61	2019-09-28 19:26:52.022	7	13	49	78	51.7001522, -0.	
74	14	52	2019-09-28 19:27:04.273	1	23	56	33	51.7001510, -0.	
62	13	82	2019-09-28 19:27:14.442	1	19	51	79	51.7001554, -0.	
85	16	66	2019-09-28 19:27:05.291	6	11	54	85	51.7001554, -0.	
49	43	74	2019-09-28 19:26:57.122	2	15	34	34	51.7001551, -0.	

Select to create a new IAM role.

When clicked on 'Discover Schema' Kinesis Analytics application will infer the schema from incoming stream, as shows here.

Cancel Save and continue

Click 'Save and continue' to move to the next step

Application ARN: arn:aws:kinesisanalytics:us-east-1:796984966265:application/windturbine_analytics_app
Application version ID: 2 ⓘ

Source

Streaming data

Connect to an existing Kinesis stream or Firehose delivery stream, or easily create and connect to a new demo Kinesis stream. Each application can connect to one streaming data source. [Learn more](#)

Source	In-application stream name	ID ⓘ	Record pre-processing ⓘ
Firehose delivery stream windturbine_input_stream ⓘ	SOURCE_SQL_STREAM_001	2.1	Disabled

Reference data (optional)

Enrich data from your streaming data source with JSON or CSV data stored as an object in Amazon S3. Each application can connect to one reference data source. [Learn more](#)

[Connect reference data](#)

Real time analytics

Author your own SQL queries or add SQL from templates to easily analyze your source data. [Learn more](#)

[Go to SQL editor](#)

Destination

(Optional) Connect an in-application stream to a Kinesis stream, or to a Firehose delivery stream, to continuously deliver SQL results to AWS destinations. The limit is three destinations for each application. [Learn more](#)

Click 'go to SQL editor'

[Exit to Kinesis Analytics applications](#)

Feedback English (US)

" or pull up the queries.

Would you like to start running "windturbine_analytics_app"?

The SQL editor is much more powerful when your application is running.

- See samples from your source data stream
- Get feedback on any errors in your configuration or SQL
- Watch as your data is processed in real-time by your SQL code

No, I'll do this later [Yes, start application](#)

temperature	temperature	COL_timestamp	wind_direction	oil_level	humidity
3	INTEGER	TIMESTAMP	INTEGER	INTEGER	INTEGER

Click to start the application

Kinesis Analytics applications > windturbine_analytics_app > SQL editor

Amazon Kinesis

Real-time analytics

Save and run SQL (highlighted with a red box)

Add SQL from templates

Download SQL

SQL reference guide

Kinesis data generator tool

```

12     oil_level      INTEGER,
13     "RPM_blade"    INTEGER,
14     "vibration_frequency" INTEGER,
15     "ANOMALY_SCORE" DOUBLE);
16
17 -- Creates an output stream and defines a schema
18 CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
19     "COL_timestamp" TIMESTAMP,
20     "geo_location" VARCHAR(32),
21     "turbine_id"    INTEGER,
22     "wind_speed"   INTEGER,

```

Copy the SQL code from '[kinesis_analytics_app_for_anomaly_detection.sql](#)', that you downloaded with this guide, and paste here. After that, click 'Save and run SQL'

Application status: RUNNING

Source data **Real-time analytics** (highlighted with a red box) Destination

In-application streams:

- DESTINATION_SQL_STREAM
- TEMP_STREAM
- error_stream

Pause results New results are added every 2-10 seconds. The results below are sampled. ⓘ

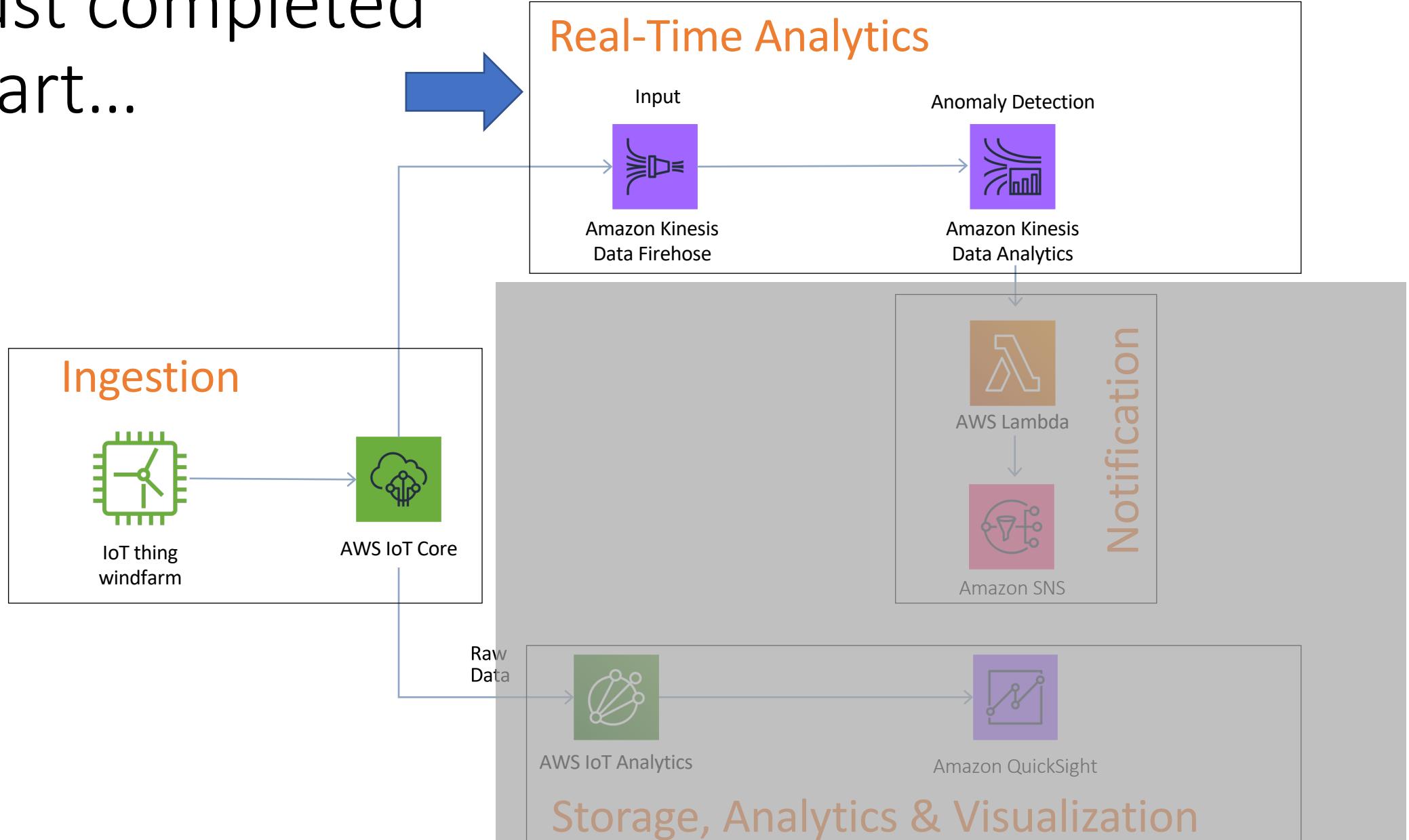
Scroll to bottom when new results arrive.

ture	temperature	wind_direction	oil_level	RPM_blade	vibration_frequency	ANOMALY_SCORE
83	3	11	70	7		0.0
62	2	10	78	12		0.0
71	8	4	70	11		0.0
71	6	22	30	7		0.0
66	3	24	63	9		0.0
41	8	13	76	15		0.0
60	4	16	68	5		0.0
68	7	19	54	13		0.0

Click on 'Real-time analytics'

Incoming records and ANOMALY_SCORE generated by the application will be shown here.

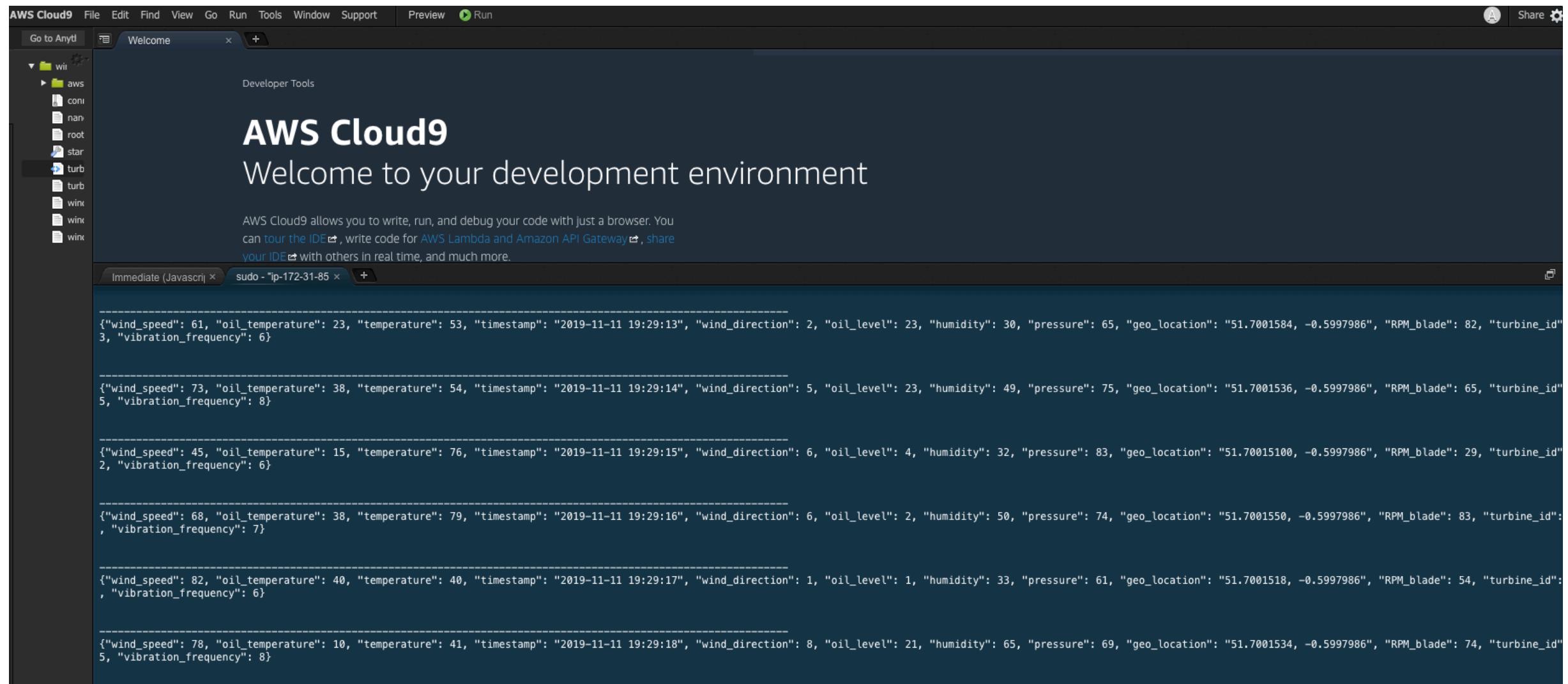
You just completed
this part...



We will go over these labs:

Lab 1 –IoT ‘thing’ & Device Simulation	15 mins
Lab 2 –Kinesis Firehose and Analytics Pipeline	30 mins
Lab 3 –IoT Analytics	30 mins
Lab 4 –Notification (Optional)	30 mins

Before setting up IoT Analytics, ensure that device simulator is running on Cloud9



The screenshot shows the AWS Cloud9 IDE interface. The top navigation bar includes File, Edit, Find, View, Go, Run, Tools, Window, Support, Preview, and Run buttons. On the right, there are icons for Share and Settings. The left sidebar shows a file tree with a folder named 'wi' containing files like aws, coni, nan, root, star, turb, turb, winc, winc, and winc. Below the sidebar is a 'Developer Tools' section. The main content area displays the AWS Cloud9 welcome message: "AWS Cloud9 Welcome to your development environment". A note below states: "AWS Cloud9 allows you to write, run, and debug your code with just a browser. You can [tour the IDE](#), write code for [AWS Lambda](#) and [Amazon API Gateway](#), share your IDE with others in real time, and much more." At the bottom, a terminal window titled "Immediate (JavaScript)" shows several JSON objects being printed:

```
{"wind_speed": 61, "oil_temperature": 23, "temperature": 53, "timestamp": "2019-11-11 19:29:13", "wind_direction": 2, "oil_level": 23, "humidity": 30, "pressure": 65, "geo_location": "51.7001584, -0.5997986", "RPM_blade": 82, "turbine_id": 3, "vibration_frequency": 6}

{"wind_speed": 73, "oil_temperature": 38, "temperature": 54, "timestamp": "2019-11-11 19:29:14", "wind_direction": 5, "oil_level": 23, "humidity": 49, "pressure": 75, "geo_location": "51.7001536, -0.5997986", "RPM_blade": 65, "turbine_id": 5, "vibration_frequency": 8}

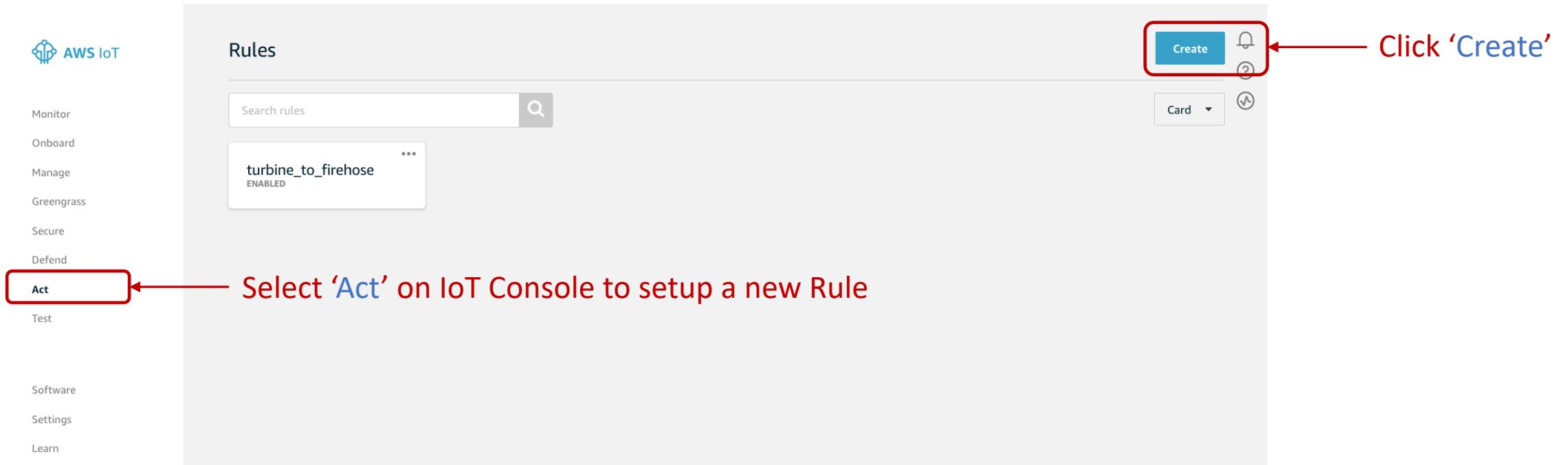
{"wind_speed": 45, "oil_temperature": 15, "temperature": 76, "timestamp": "2019-11-11 19:29:15", "wind_direction": 6, "oil_level": 4, "humidity": 32, "pressure": 83, "geo_location": "51.70015100, -0.5997986", "RPM_blade": 29, "turbine_id": 2, "vibration_frequency": 6}

{"wind_speed": 68, "oil_temperature": 38, "temperature": 79, "timestamp": "2019-11-11 19:29:16", "wind_direction": 6, "oil_level": 2, "humidity": 50, "pressure": 74, "geo_location": "51.7001550, -0.5997986", "RPM_blade": 83, "turbine_id": 5, "vibration_frequency": 7}

 {"wind_speed": 82, "oil_temperature": 40, "temperature": 40, "timestamp": "2019-11-11 19:29:17", "wind_direction": 1, "oil_level": 1, "humidity": 33, "pressure": 61, "geo_location": "51.7001518, -0.5997986", "RPM_blade": 54, "turbine_id": 5, "vibration_frequency": 6}

 {"wind_speed": 78, "oil_temperature": 10, "temperature": 41, "timestamp": "2019-11-11 19:29:18", "wind_direction": 8, "oil_level": 21, "humidity": 65, "pressure": 69, "geo_location": "51.7001534, -0.5997986", "RPM_blade": 74, "turbine_id": 5, "vibration_frequency": 8}
```

Setup IoT rule to push data in IoT Analytics



Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name turbo

Description

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

Using SQL version

Rule query statement
SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see [AWS IoT SQL Reference](#).

```
1 SELECT * FROM 'iot/topic'
```

Set one or more actions
Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*.required)

[Add action](#)

Give a name to this rule.

Change the SQL code here to `SELECT * FROM 'topic_1'`

Click 'Add action' to link this rule to IoT Analytics

-  Send a message to an SQS queue
SQS
-  Send a message to an Amazon Kinesis Stream
AMAZON KINESIS
-  Republish a message to an AWS IoT topic
AWS IOT PUBLISH
-  Store a message in an Amazon S3 bucket
S3
-  Send a message to an Amazon Kinesis Firehose stream
AMAZON KINESIS FIREHOSE
-  Send message data to CloudWatch
CLOUDWATCH METRICS
-  Change the state of a CloudWatch alarm
CLOUDWATCH ALARMS
-  Send a message to the Amazon Elasticsearch Service
AMAZON ELASTICSEARCH
-  Send a message to a Salesforce IoT Input Stream
SALESFORCE IOT
-  Send a message to IoT Analytics
IOT ANALYTICS
-  Send a message to an IoT Events Input
IOT EVENTS
-  Start a Step Functions state machine execution
STEP FUNCTIONS

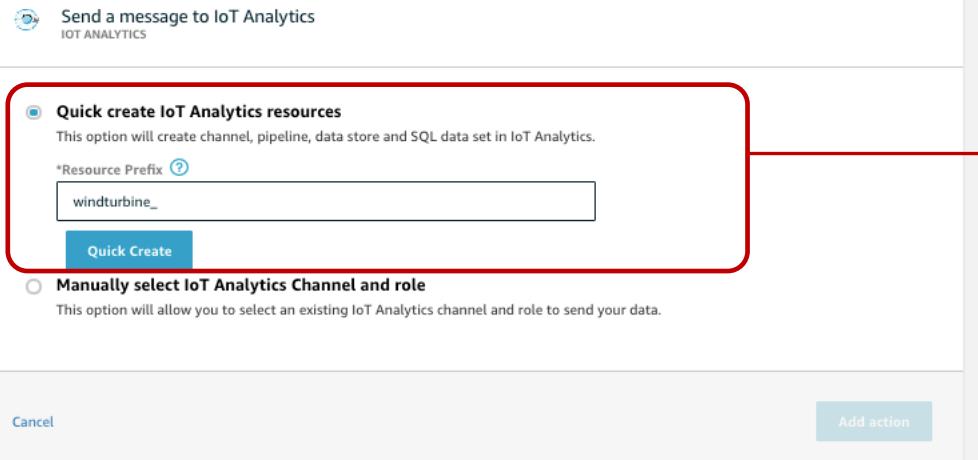
Cancel

Configure action

Select 'IoT Analytics' from list of services

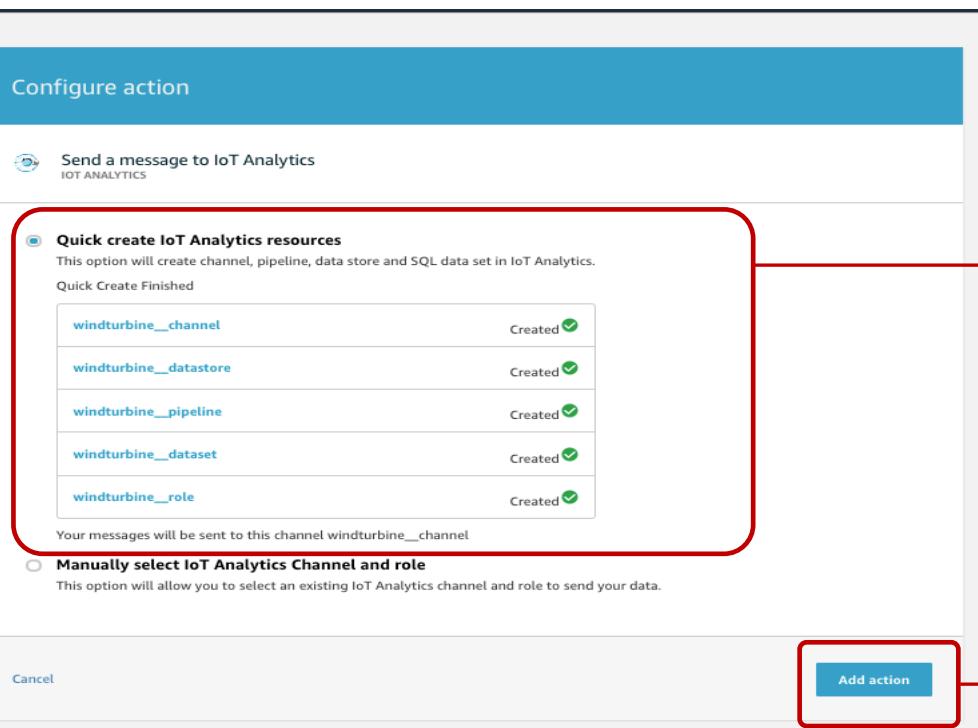
Click 'Configure Action'

Configure action



The IoT Analytics resources do not exist at this point. We will select the 'Quick Create IoT Analytics option. Enter 'windturbine' as the resource prefix, then click 'Quick Create'

Configure action



After clicking 'Quick Create' you will see the IoT Analytics resources created.

Click 'Add action'

learn more, see [AWS IoT SQL Reference](#).

```
1 | SELECT * FROM 'topic_1'
```

Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*.required)

 Send a message to IoT Analytics
windturbine__channel Remove Edit >

[Add action](#)

Error action

Optionally set an action that will be executed when something goes wrong with processing your rule.

[Add action](#)

Tags

Apply tags to your resources to help organize and identify them. A tag consists of a case-sensitive key-value pair. [Learn more](#) about tagging your AWS resources.

Tag name Value Clear

[Add another](#)

[Cancel](#) Create rule



Click 'Create Rule' to complete this process.

Transforming and querying data in IoT Analytics



Channels

Name	Status
windturbine_channel	ACTIVE



Pipelines

Name	Created
windturbine_pipeline	Nov 11, 2019 11:44:26

Launch AWS IoT Analytics from AWS Console.

You will see the Channel, Pipeline and Data Store and Data set created from previous step

In this lab we will use pipeline to create a simple transformation – converting temperature from Fahrenheit to Celsius

We will start with first selecting the pipeline

PIPELINE

windturbine_pipeline

Actions ▾

Overview Channel inputs Details

Name	Type	Edit
windturbine_channel	Channel	

Activities

Name	Type	Edit

Data store outputs

Name	Type	Edit
windturbine_datastore	Data store	

Tags

No tags	Edit

Click on 'Edit' Activity, to add a transformation step.

EDIT PIPELINE
windturbine_pipeline STEP 1/2

Pipelines enrich, transform, and filter messages based on their attributes. Upload a sample JSON message or enter attributes manually to get started.

Attributes

Action ▾

Attribute name	
<input type="checkbox"/> wind_speed	57
<input type="checkbox"/> oil_temperature	26
<input checked="" type="checkbox"/> temperature	71
<input type="checkbox"/> timestamp	"2019-11-11 19:48:26"
<input type="checkbox"/> wind_direction	6
<input type="checkbox"/> oil_level	6
<input type="checkbox"/> humidity	40
<input type="checkbox"/> pressure	49
<input type="checkbox"/> geo_location	"51.7001590, -0.5997986"
<input type="checkbox"/> RPM_blade	31
<input type="checkbox"/> turbine_id	7
<input type="checkbox"/> vibration_frequency	5

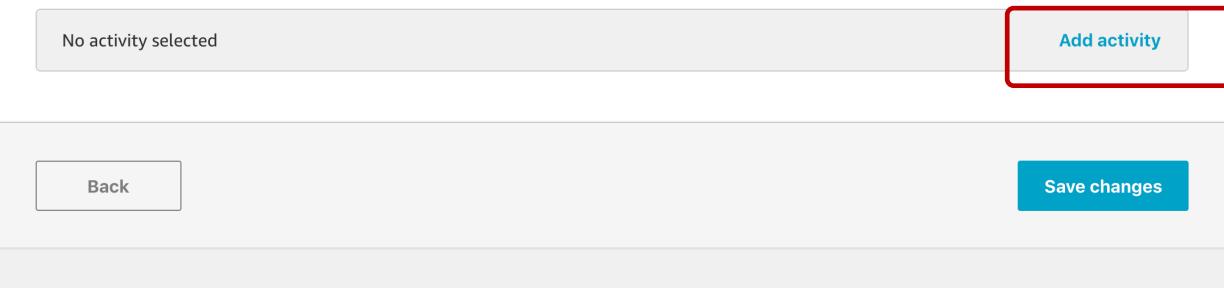
Cancel Next

Select 'temperature' attribute



Pipeline activities

Chaining activities together enables you to process and prepare messages before storing them. You can enrich or transform message attributes, or filter entire messages out of your pipeline.



Click Add activity

EDIT PIPELINE

windturbine_pipeline

STEP 2/2

Pipeline activities

Chaining activities together enables you to process and prepare messages before storing them. You can enrich or transform message attributes, or filter entire messages out of your pipeline.

No activity selected		Close
Remove attributes from the message	TRANSFORM	
Select attributes from the message	TRANSFORM	
Add attributes to the message	TRANSFORM	
Calculate a message attribute	TRANSFORM	

Back

Save changes

Select Calculate a message attribute

• Pipeline activities

Chaining activities together enables you to process and prepare messages before storing them. You can enrich or transform message attributes, or filter entire messages out of your pipeline.

The screenshot shows a 'Calculate a message attribute' step in a pipeline. At the top, there's a toolbar with a gear icon, the step name, 'TRANSFORM', and 'Remove Close' buttons. Below the toolbar is a section titled 'Incoming messages' containing a list of attributes and their values:

- wind_speed 57
- oil_temperature 26
- temperature 71
- timestamp "2019-11-11 19:48:26"
- wind_direction 6
- oil_level 6
- humidity 40
- pressure 49
- geo_location "51.7001590,-0.5997986"
- RPM_blade 31
- turbine_id 7
- vibration_frequency 5

Below this is a 'Calculate a message attribute' section with the following fields:

- Attribute name:** temperature_C
- Formula:** $(\text{temperature} - 32) * 5 / 9$

At the bottom, there's an 'Outgoing message' section with the text: 'Below is a list of attributes to be included in the outgoing message.'

In this step, we convert the temperature to Celsius, and add an additional attribute 'temperature_c'

Select 'Calculate a message attribute'.
The formula here is $(\text{temperature} - 32) * 5 / 9$

Attribute name	Formula
temperature_C	(temperature - 32) * 5 / 9

Outgoing message

Below is a list of attributes to be included in the outgoing message.

[Update preview](#) [Load a new message](#)

wind_speed 57

oil_temperature 26

temperature 71

timestamp "2019-11-11 19:48:26"

wind_direction 6

oil_level 6

humidity 40

pressure 49

geo_location "51.7001590,-0.5997986"

RPM_blade 31

turbine_id 7

vibration_frequency 5

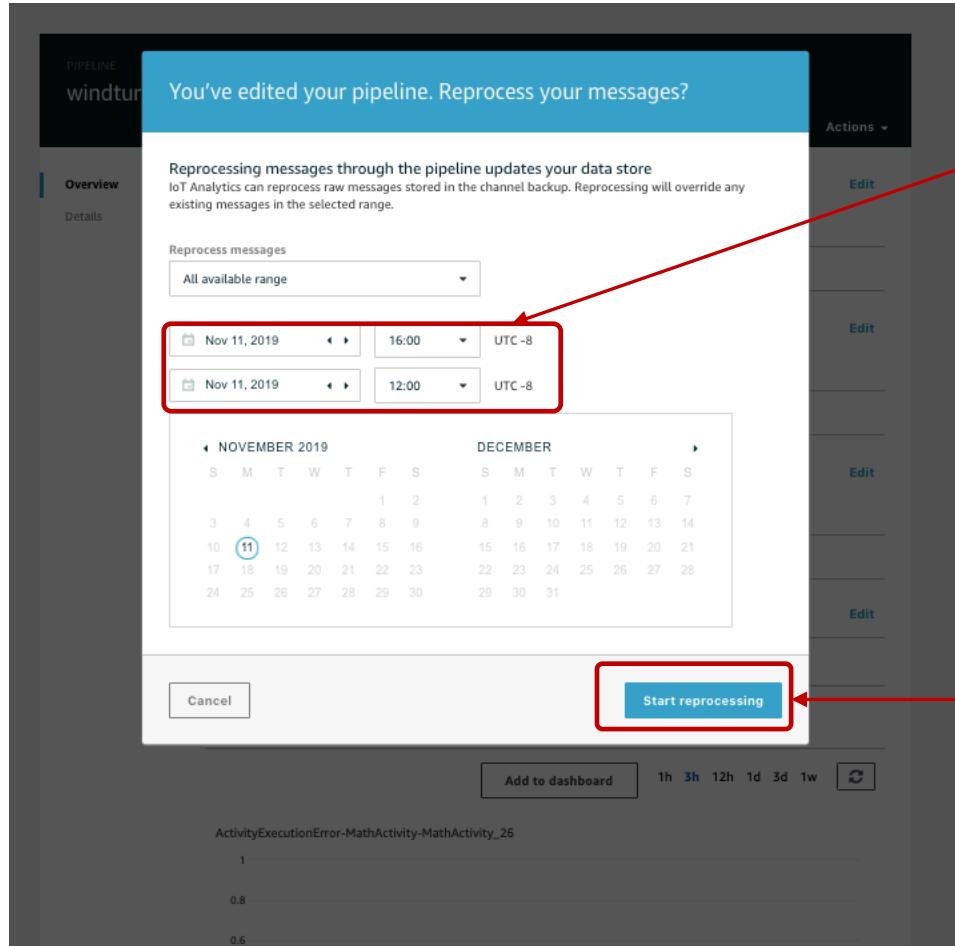
No activity selected

Add activity

Back

Save changes

Save changes to apply activity to this pipeline



You will see an option to reprocess the messages. You can choose a timeframe to go back and apply this transformation to previously collected data.

Click 'Start processing'

On the IoT Analytics console, select 'Data sets' from the left panel



Channels
Pipelines
Data stores
Data sets
Notebooks

Data sets

Name	Type
windturbine_dataset	Query

Click on the Data set 'Windturbine_dataset', you will open the screen below.

DATA SET
windturbine_dataset
SUCCEEDED Actions ▾

Details Data set ARN
Content A data set Amazon Resource Name (ARN) uniquely identifies this data set.
`arn:aws:iotanalytics:us-east-1:796984966265:dataset/windturbine_dataset`

Details

SQL query Edit
`select * from windturbine_dataset`

Delta window Edit
Delta window has not been set yet.

Result preview

wind_speed	oil_temperature	temperature	timestamp	wind_direction
84	14	81	2019-11-11 19:51:00	5
68	40	64	2019-11-11 19:51:01	1
83	11	59	2019-11-11 19:51:02	6
70	38	45	2019-11-11 19:51:03	3
81	47	41	2019-11-11 19:51:04	1

Creation date Nov 11, 2019 11:44:26 AM -0800

Last updated date Nov 11, 2019 11:44:26 AM -0800

Schedule Add schedule

Here you can preview incoming transformed data

Click 'Add schedule' to configure frequency to run this query. You can set frequency between 1 minute and a month to refresh the datasets.

For this workshop, we can set it for 1 minute, so we can visualize this data in later steps.

Visualize IoT data in AWS QuickSight

Select Quicksight from AWS Console

If QuickSight is not setup, you will see the following screen.

Your AWS Account is not signed up for QuickSight. Would you like to sign up now?

AWS Account 796984966265

[Sign up for QuickSight](#)

To access QuickSight with a different account, [log in](#) again.

Create your QuickSight account

Edition Standard Enterprise

	FREE	FREE
First author with 1GB SPICE	FREE	FREE
Team trial for 60 days (4 authors)*	FREE	FREE
Additional author per month (yearly)**	\$9	\$18
Additional author per month (monthly)**	\$12	\$24
Additional readers (Pay-per-Session)	N/A	\$0.30/session (max \$5/reader/month) ****
Additional SPICE per month	\$0.25 per GB	\$0.38 per GB
Single Sign On with SAML or OpenID Connect	✓	✓
Connect to spreadsheets, databases & business apps	✓	✓
Access data in Private VPCs		✓
Row-level security for dashboards		✓
Hourly refresh of SPICE data		✓
Secure data encryption at rest		✓
Connect to your Active Directory		✓
Use Active Directory Groups ***		✓
Send email reports		✓

* Trial authors are auto-converted to month-to-month subscription upon trial expiry
** Each additional author includes 1GB of SPICE capacity
*** Active Directory groups are available in accounts connected to Active Directory
**** Sessions of 30-minute duration. Total charges for each reader are capped at \$5 per month. [Conditions apply](#)

[Continue](#)

Click 'Sign up for QuickSight'

Select Standard

Click Continue

Edition Standard

QuickSight region

Select a region.

US East (N. Virginia)

QuickSight account name

iot-demo

You will need this for you and others to sign in.

Notification email address

r...zon.com

For QuickSight to send important notifications.

Enable autodiscovery of data and users in your Amazon Redshift, Amazon RDS, and AWS IAM services.

Amazon Athena

Enables QuickSight access to Amazon Athena databases

Please ensure the right Amazon S3 buckets are also enabled for QuickSight.

Amazon S3

Enables QuickSight to auto-discover your Amazon S3 buckets

Amazon S3 Storage Analytics

Enables QuickSight to visualize your S3 Storage Analytics data

AWS IoT Analytics

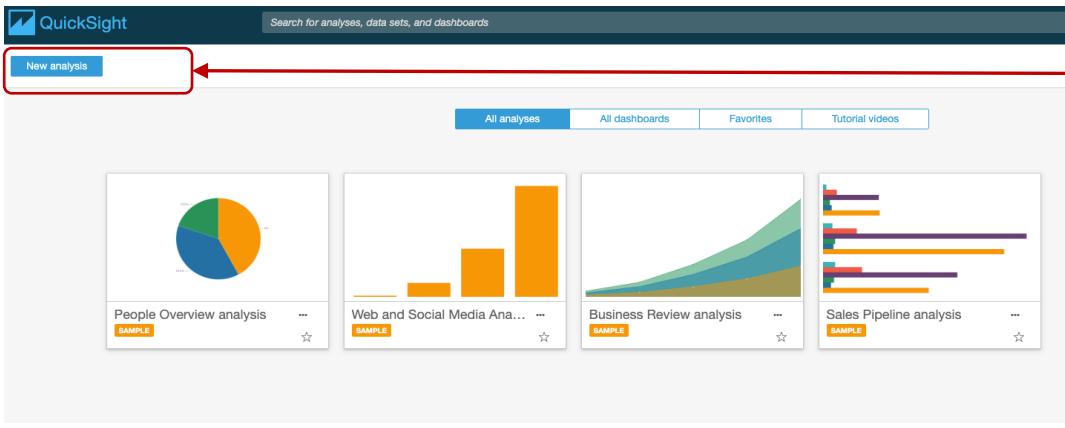
Enables QuickSight to visualize your IoT Analytics data

Choose S3 buckets

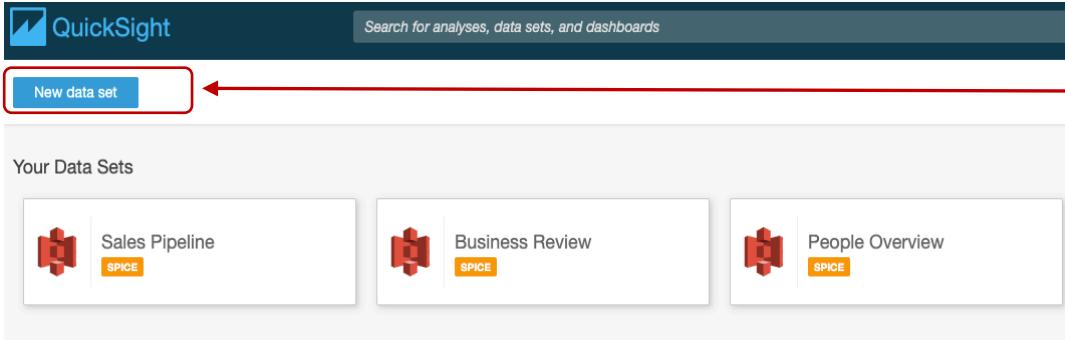
Finish

Provide a unique name and email-id

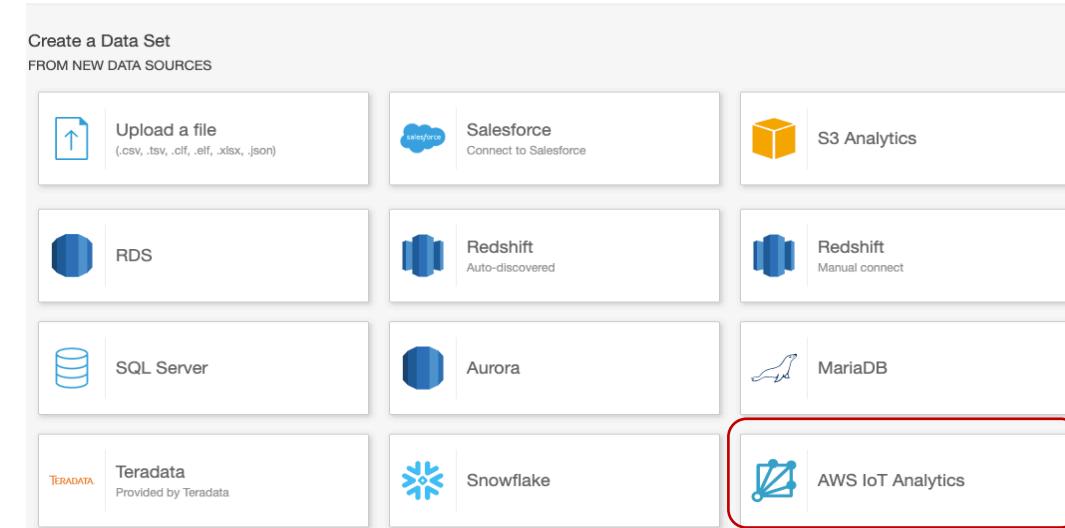
Select AWS IoT Analytics, to give permission to QuickSight, to access IoT Analytics datasets



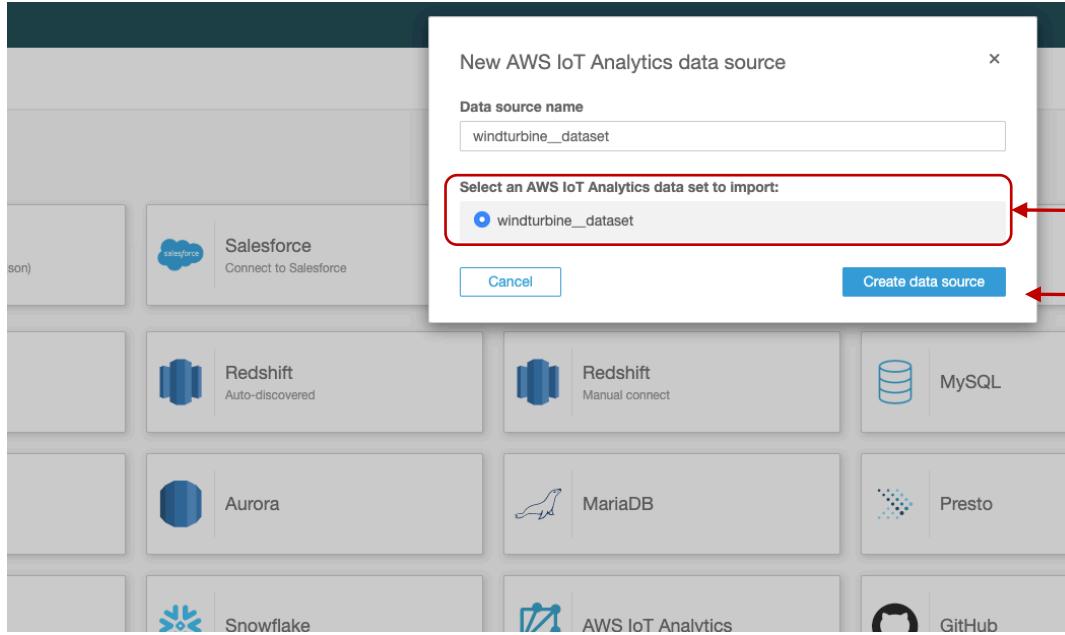
Click 'New Analysis'



Click 'New Data Set'



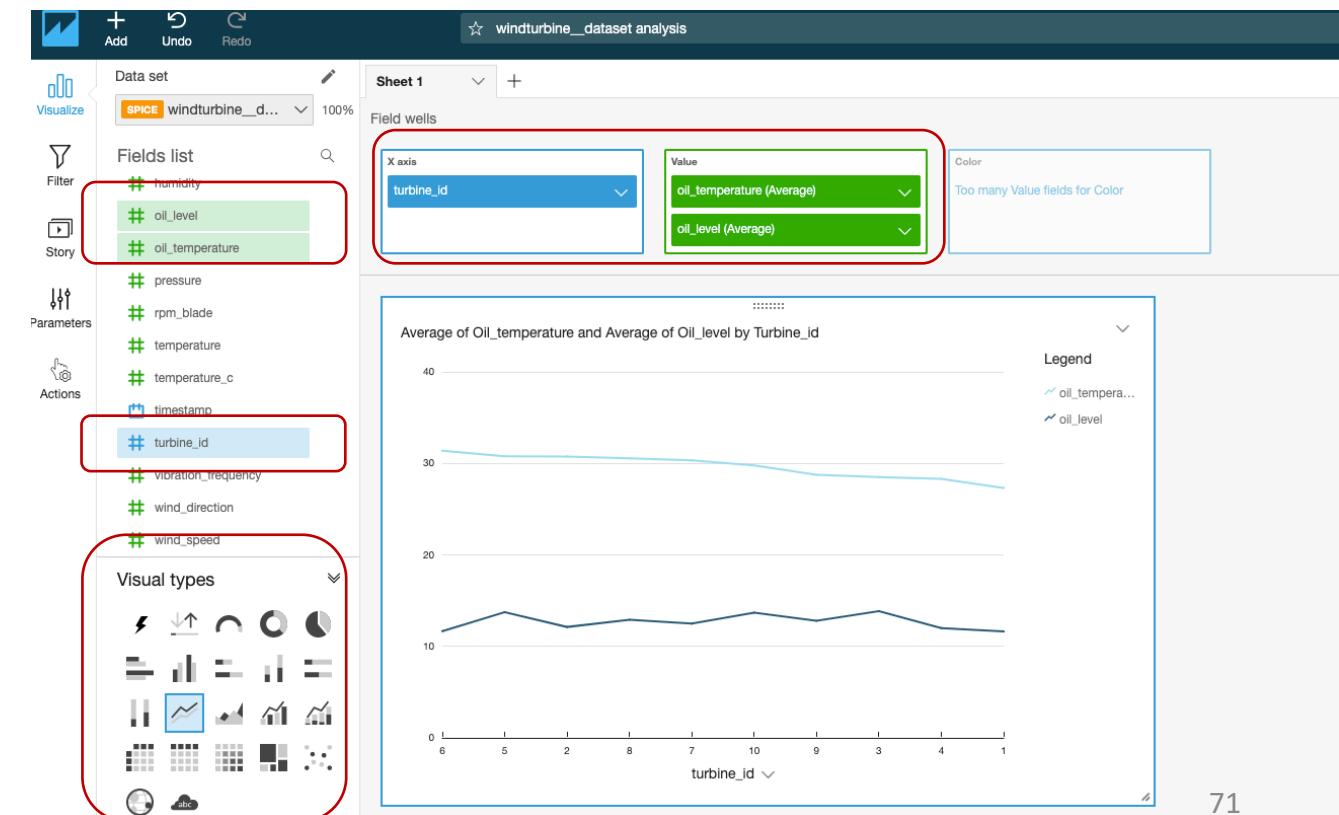
Select 'AWS IoT Analytics'



Select the Dataset.

Click 'Create data source'

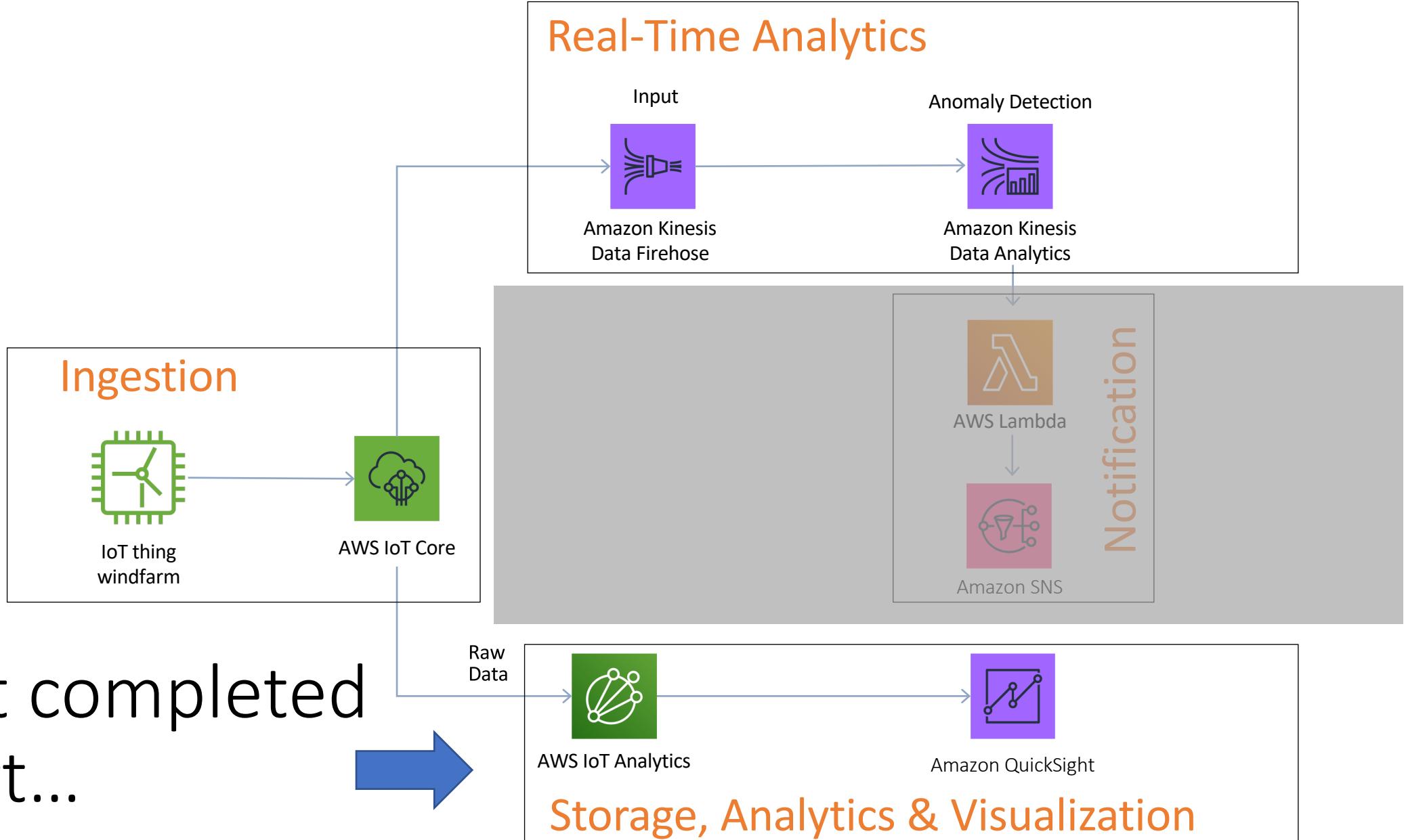
Next, start visualizing the data



In this example, under Visual Types on the bottom left, select Line Graph.

Then choose `turbine_id` to be the X-axis and temperature and `oil_temperature` to be the Y-axis.

In the drop down of temperature, choose the 'Aggregate' option and select 'Average'. Do the same with `oil_temperature`



We will go over these labs:

Lab 1 –IoT ‘thing’ & Device Simulation 15 mins

Lab 2 –Kinesis Firehose and Analytics Pipeline 30 mins

Lab 3 –IoT Analytics 30 mins

Lab 4 –Notification (Optional) 30 mins

The screenshot shows the AWS SNS landing page. At the top, there's a navigation bar with 'Services', 'Resource Groups', and other account details. Below the header, the main title is 'Amazon Simple Notification Service' with the subtitle 'Pub/sub messaging for microservices and serverless applications.' A descriptive paragraph follows, mentioning its availability and use cases. On the left, there's a section titled 'Benefits and features' with four items: 'Reliably deliver messages with durability', 'Automatically scale your workload', 'Simplify your architecture with Message Filtering', and 'Keep messages private and secure'. On the right, there are sections for 'Pricing', 'Documentation' (with links to Developer Guide, API Reference, FAQs, and Support forums), and 'Explore AWS' (with a link to Amazon RDS). At the bottom, there's a 'Use cases' section featuring logos for NASA and PlayOn! SPORTS. The bottom navigation bar includes 'Feedback', 'English (US)', and links to 'Privacy Policy' and 'Terms of Use'.

Create topic

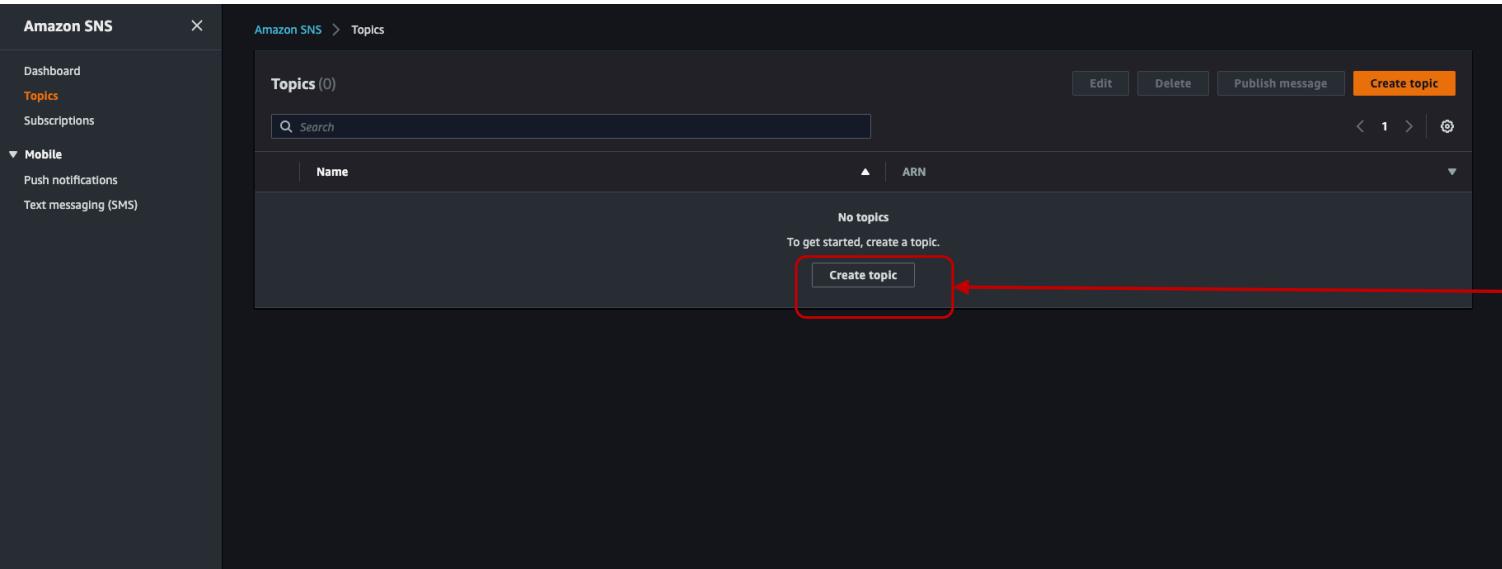
Topic name: turbine_anomaly_notification

Next step

Start with an overview

Select SNS from AWS Console.

Click 'Next step'



Click 'Create topic'

The screenshot shows the 'Create topic' form. At the top left, it says 'Amazon SNS > Topics > Create topic'. The main section is titled 'Create topic' and has a 'Details' tab selected. The first field is 'Name', which contains the value 'turbine_anomaly_notification'. A red box highlights this input field, and a red arrow points from the text 'Give a name to this topic. Leave all other settings default, then click 'Create topic'' to it. Below the 'Name' field is a note: 'Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).'. The next section is 'Display name - optional', which contains a field 'My Topic' with the value 'My Topic'. A note below it says: 'To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.' The 'Info' link is shown in blue. A note at the bottom of this section says: 'Maximum 100 characters, including hyphens (-) and underscores (_).'. There are three expandable sections: 'Encryption - optional', 'Access policy - optional', and 'Delivery retry policy (HTTP/S) - optional'. Each section has a note at the bottom: 'Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.', 'This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic.', and 'The policy defines how Amazon SNS retries failed deliveries to HTTP/S endpoints. To modify the default settings, expand this section.' respectively. The 'Info' links for each section are also shown in blue.

Give a name to this topic. Leave all other settings default, then click 'Create topic'

Note: After creating subscription, you will receive an email to confirm the subscription. SNS will not send notification, unless you confirm the subscription.

The screenshot shows the 'Subscriptions' tab for the 'turbine_anomaly_notification' topic. The 'Create subscription' button is highlighted with a red box and an arrow pointing to it from the text 'Click 'Create subscription''. The 'Protocol' dropdown is set to 'Email' and the 'Endpoint' field contains '@amazon.com'. A note at the bottom states: 'After your subscription is created, you must confirm it.' A red box highlights the 'Create subscription' button at the bottom right of the modal.

Select 'Email' Protocol

Provide your email-id as endpoint

Click 'create subscription'

Click 'Create subscription'

Amazon SNS

turbine_anomaly_notification

Edit Delete Publish message

Dashboard

Topics

Subscriptions

Mobile

Push notifications

Text messaging (SMS)

Details

Name: turbine_anomaly_notification

Display name: -

ARN: arn:aws:sns:us-east-1:79[REDACTED]65:turbine_anomaly_notification

Topic owner: 79[REDACTED]65

Subscriptions Access policy Delivery retry policy (HTTP/S) Delivery status logging Encryption Tags

Subscriptions (1)

Edit Delete Request confirmation Confirm subscription Create subscription

Search < 1 > ⚙️

ID	Endpoint	Status	Protocol
890bd1dd-1[REDACTED]066318cf	[REDACTED]@amazon.com	Confirmed	EMAIL



Before proceeding, ensure the status is shown as 'Confirmed'.

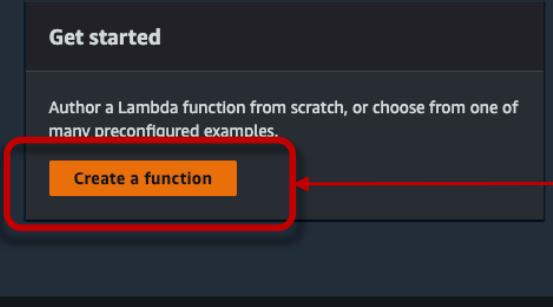
Create Lambda Function

COMPUTE

AWS Lambda

lets you run code without thinking about servers.

You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.



Select Lambda from AWS Console. Click 'Create a function'

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The first step, 'Choose one of the following options to create your function.', is displayed. Three options are available: 'Author from scratch' (with a gear icon), 'Use a blueprint' (selected, indicated by a blue dot and highlighted with a red box), and 'Browse serverless app repository' (with a cloud and file icon). The 'Blueprints' section below shows a search bar with 'kinesis-analytics' typed in, and a list of available blueprints. One blueprint, 'kinesis-analytics-output-sns', is highlighted with a red box. At the bottom right of the blueprint list is a 'Configure' button, also highlighted with a red box.

Choose one of the following options to create your function.

Author from scratch
Start with a simple Hello World example.
Blueprints Info

Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

Browse serverless app repository
Deploy a sample Lambda application from the AWS Serverless Application Repository.

Blueprints Info

Keyword : kinesis-analytics

kinesis-analytics-output-sns
Deliver output records from Kinesis Analytics application to a SNS topic.
python2.7 · kinesis-analytics

kinesis-analytics-process-kpl-records
An Amazon Kinesis Analytics record pre-processor that receives Kinesis Producer Library (KPL) aggregates of JSON or CSV records as input and returns de-aggregated records with a processing status.
python2.7 · kinesis-analytics

kinesis-analytics-output-ddb
Deliver output records from Kinesis Analytics application to a DynamoDB table.
python2.7 · kinesis-analytics

Configure

Choose Blueprint

Search for 'kinesis-analytics'

Select 'kinesis-analytics-output-sns' blueprint

Click 'configure'

**Basic information** Info**Function name**

anomaly-trigger

Execution roleChoose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

Create a new role from AWS policy templates

Give function a name – ‘anomaly-trigger’

Create a new IAM role, to enable Lambda to push messages to SNS

i Role creation might take a few minutes. The new role will be scoped to the current function. To use it with other functions, you can modify it in the IAM console.

Role nameEnter a name for your new role.

lambda-sns-push-role

Use only letters, numbers, hyphens, or underscores with no spaces.**Policy templates** InfoChoose one or more policy templates.

Amazon SNS publish policy X

SNS

Lambda function codeCode is preconfigured by the chosen blueprint. You can configure it after you create the function. [Learn more](#) about deploying Lambda functions.

Lambda function code

Code is preconfigured by the chosen blueprint. You can configure it after you create the function. [Learn more](#) about deploying Lambda functions.

Runtime

Python 2.7

```
1 from __future__ import print_function
2 import boto3
3 import base64
4
5 client = boto3.client('sns')
6 # Include your SNS topic ARN here.
7 #topic_arn = 'arn:aws:sns:<region>:<account_id>:<topic_name>'
8
9
10 def lambda_handler(event, context):
11     output = []
12     success = 0
13     failure = 0
14     for record in event['records']:
15         try:
16             # Uncomment the below line to publish the decoded data to the SNS topic.
17             #payload = base64.b64decode(record['data'])
18             #client.publish(TopicArn=topic_arn, Message=payload, Subject='Sent from'
19             #output.append({'recordId': record['recordId'], 'result': 'Ok'})
20             success += 1
21         except Exception:
22             output.append({'recordId': record['recordId'], 'result': 'DeliveryFailed'})
23             failure += 1
24
25     print('Successfully delivered {} records, failed to deliver {} records'.format(
26         len(output), failure))
27     return {'records': output}
```

Cancel

Create function

Click 'Create function'

The screenshot shows the AWS Lambda console's code editor. At the top, it displays "Code entry type: Edit code inline", "Runtime: Python 2.7", and "Handler: lambda_function.lambda_handler". The main area contains Python code for a Kinesis Analytics function. A red box highlights the line "#topic_arn = 'arn:aws:sns:<region>:<account_id>:<topic_name>'". Another red box highlights the commented-out SNS publishing logic starting with "# Uncomment the below line to publish the decoded data to the SNS topic.". The code editor interface includes tabs for "Environment" and "lambda_function", and a file tree on the left showing "anomaly-trigger" and "lambda_function.py".

```
from __future__ import print_function
import boto3
import base64
client = boto3.client('sns')
# Include your SNS topic ARN here.
topic_arn = 'arn:aws:sns:<region>:<account_id>:<topic_name>'

def lambda_handler(event, context):
    output = []
    success = 0
    failure = 0
    for record in event['records']:
        try:
            # Uncomment the below line to publish the decoded data to the SNS topic.
            #payload = base64.b64decode(record['data'])
            #client.publish(TopicArn=topic_arn, Message=payload, Subject='Sent from Kinesis Analytics')
            output.append({'recordId': record['recordId'], 'result': 'Ok'})
            success += 1
        except Exception:
            output.append({'recordId': record['recordId'], 'result': 'DeliveryFailed'})
            failure += 1
    print('Successfully delivered {} records, failed to deliver {} records'.format(success, failure))
    return {'records': output}
```

On the Lambda console, scroll down to the function code.

The function code is already filled in since we selected an existing blueprint. We need to add a few things before making it functional

Add the SNS ARN that we created earlier, and uncomment this line.

Uncomment the line here, when you are ready to send SNS message.

For reference, the entire lambda code is on the next page, you can copy paste and replace the default code in Lambda console.

You can copy past the Lambda code from here and replace the existing lambda function code.
Don't forget to add your SNS topic ARN in the code! This code is also available in `lambda_code.py` you downloaded with the workshop package

```
from __future__ import print_function
import boto3
import base64
from json import loads

client = boto3.client('sns')
# Include your SNS topic ARN here.
topic_arn = 'Add your SNS topic SRN here'

def lambda_handler(event, context):
    output = []
    success = 0
    failure = 0
    for record in event['records']:
        try:

            payload = base64.b64decode(record['data'])
            data_item = loads(payload)
            #print (data_item)

            if (data_item['ANOMALY_SCORE']) > 2:
                client.publish(TopicArn=topic_arn, Message=payload, Subject='Anomaly Detected!')
                print ('Anomaly Detected')
                output.append({'recordId': record['recordId'], 'result': 'Ok'})
                success += 1
            except Exception:
                output.append({'recordId': record['recordId'], 'result': 'DeliveryFailed'})
                failure += 1

    print('Successfully delivered {0} records, failed to deliver {1} records'.format(success, failure))
    return {'records': output}
```

Link Lambda function to Kinesis Analytics App

Amazon Kinesis dashboard

Amazon Kinesis

Dashboard

Data Streams

Data Firehose

Data Analytics

Video Streams

External resources

What's new

Amazon Kinesis makes it easy to collect, process, and analyze video and data streams in real time, so you can get timely insights and react quickly to new information. [What is streaming data?](#)

Kinesis data streams

Process data with your own applications, or using AWS managed services like Amazon Kinesis Data Firehose, Amazon Kinesis Data Analytics, or AWS Lambda. [Learn more](#)

[Create data stream](#)

Kinesis Firehose delivery streams (2)

Name	Status
windturbine_input_stream	Active
windturbine_output_stream	Active

[View all](#) [Create delivery stream](#)

Kinesis analytics applications (1)

Name	Status
windturbine_analytics_app	Running

[View all](#) [Create analytics application](#)

Kinesis video streams

Build applications to process or analyze streaming media. [Learn more](#)

[Create video stream](#)

Go back to the Kinesis dashboard and select Kinesis Analytics App. We will connect a new destination to the app to send data to a Lambda Function. The Lambda Function will trigger SNS notification, when Anomaly is identified.

Video Streams

External resources

What's new

Streaming data

Connect to an existing Kinesis stream or Firehose delivery stream, or easily create and connect to a new demo Kinesis stream. Each application can connect to one streaming data source. [Learn more](#)

Source	In-application stream name	ID	Record pre-processing
Firehose delivery stream windturbine_input_stream	SOURCE_SQL_STREAM_001	2.1	Disabled

[Connect reference data](#)

Real time analytics

Continuously analyzing your source data with SQL. [Learn more](#)

[Go to SQL results](#)

Destination

(Optional) Connect an in-application stream to a Kinesis stream, or to a Firehose delivery stream, to continuously deliver SQL results to AWS destinations. The limit is three destinations for each application.

[Connect new destination](#) [Disconnect destination](#)

Destination	In-application stream name	ID
Firehose delivery stream windturbine_output_stream	DESTINATION_SQL_STREAM	7.1

Click, 'Connect new destination'

Kinesis Analytics applications > windturbine_analytics_app > Destination

Amazon Kinesis

- Dashboard
- Data Streams
- Data Firehose
- Data Analytics**
- Video Streams
- External resources
- What's new

Connect to destination

Destination*

- Kinesis stream i
- Kinesis Firehose delivery stream i
- AWS Lambda function i

Select Lambda as destination

Deliver records to AWS Lambda

To deliver Kinesis Analytics output records, your Lambda function must be compliant with the required request/response model. [Learn more](#)

Lambda function* ▼
[View anomaly-trigger in Lambda](#) ↗

Select the Lambda function we created in previous step

Lambda function version* ▼

Description Deliver output records from Kinesis Analytics application to a SNS topic.

Runtime python2.7

Increase Lambda function timeout
 To reduce the risk of the function timing out, increase the **Timeout** to 1 minute or longer in the **Advanced settings** section of your Lambda configuration.
[Go to Lambda configuration](#) ↗

Timeout 3 seconds

In-application stream

In-application streams are continuous flows of data records. You create in-application streams in SQL to contain the data you want to persist to the specified destination. [Learn more](#).

Connect in-application stream

- Choose an existing in-application stream
- Specify a new in-application stream name
 Use this option for in-application streams that you haven't created yet, but plan to create at a later time. Specifying a stream name ensures that you don't lose output data.

Select In-application Stream 'DESTINATION_SQL_STREAM_Lambda'

In-application stream name* ▼

Select 'JSON' output format

Output format

- JSON
- CSV

Access permissions

Create or choose IAM role with the required permissions. [Learn more](#)

Access permissions*

- Create / update IAM role kinesis-analytics-windturbine_analytics_app-us-east-1
- Choose from IAM roles that Kinesis Analytics can assume

Click Save and continue

When anomalies are detected, SNS will send a notification to the endpoint (Email) you specified earlier.

{"COL_timestamp":"2019-11-15 22:15:26.000","geo_location":"51.7001597,-0.5997986","turbine_id":1,"wind_speed":1,"oil_temperature":1,"temperature":1,"wind_direction":1,"oil_level":1,"RPM_blade":1,"vibration_frequency":1,"ANOMALY_SCORE":3.1708571428571433}

--

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:4244:turbine_anomaly_detection:3cd0e009&Endpoint=:amazon.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>

This is what you built today...

