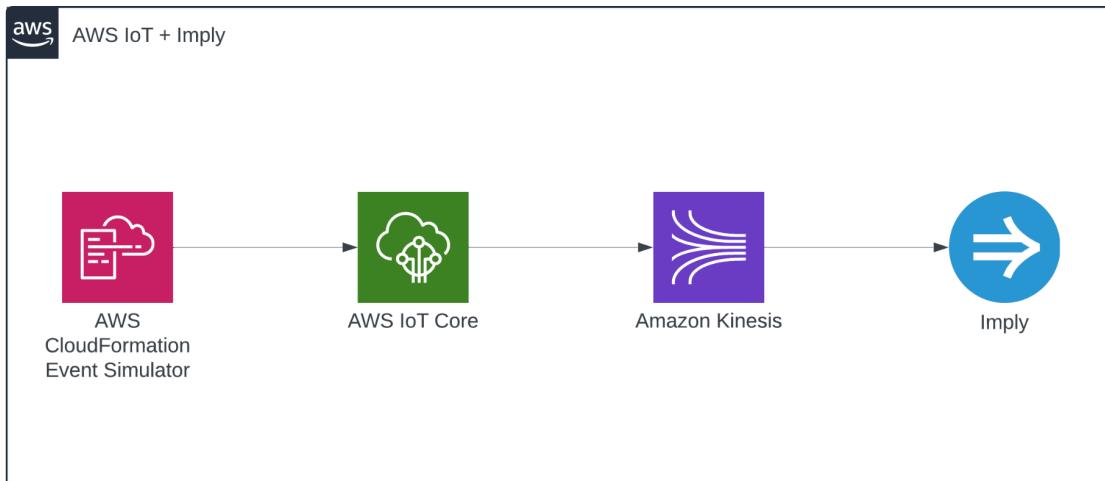


IoT Events Workshop | End-to-End Setup

The following workshop will take you step-by-step through setting up an easy data generation tool, publishing the data to AWS IoT Core, streaming the data to Kinesis, ingesting the event data to Imply Polaris, and then engineering data analytics solutions.

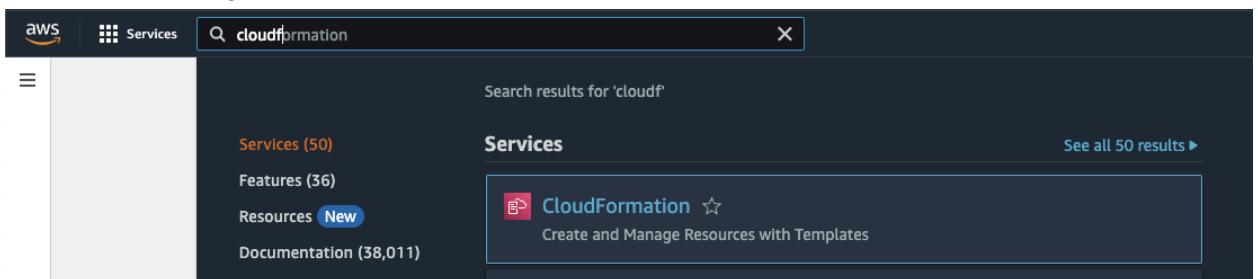


Step 0 - Prerequisites

1. Make sure you have permissions to manage and create resources on your AWS project
2. Follow the steps to setup an [Imply Polaris free trial](#) account and sign-in for the first time
 - a. Select a region for the Imply project that ideally match (this is not mandatory)
 - b. us-east-1 is a good one to use for the workshop

Step 1 - Setup of the data simulator

1. After signing into your AWS environment, search for “CloudFormation”, here you will create a stack to generate data.



- Click the “Create stack” button, your page should look like this

The screenshot shows the 'Create stack' wizard in the AWS CloudFormation console. The current step is 'Prerequisite - Prepare template'. Under 'Prepare template', the 'Template is ready' radio button is selected. There are other options: 'Use a sample template' and 'Create template in Designer'. Below this, the 'Specify template' step is shown, which includes a 'Template source' section with 'Amazon S3 URL' selected and a URL entered. There are also options for 'Upload a template file' and 'Sync from Git - new'. At the bottom right are 'View in Designer', 'Cancel', and 'Next' buttons.

- Download the latest data generator template from here by clicking the ‘View template’ button, we will upload this template to the stack we are creating, then click “Next” <https://docs.aws.amazon.com/solutions/latest/iot-device-simulator/template.html>
- Click the “Upload a template file” option and then “choose file”, picking the template we just downloaded from the previous step, then click “Next” and

The screenshot shows the 'Create stack' wizard in the AWS CloudFormation console. The current step is 'Prerequisite - Prepare template'. Under 'Prepare template', the 'Template is ready' radio button is selected. Below it, the 'Specify template' step is shown with the 'Template source' section. The 'Upload a template file' radio button is selected, and a file named 'iot-device-simulator.template' is chosen. The 'S3 URL' field shows the path 'https://s3.us-east-1.amazonaws.com/cf-templates-11ksm8bmcvpxg-us-east-1/2024-03-04T172504.799Z/iot-device-simulator.template'. At the bottom right are 'View in Designer', 'Cancel', and 'Next' buttons.

- Specify your “Stack name”, then enter your email address for the “Console Administrator Email”, then click “Next”

CloudFormation > Stacks > Create stack

Specify stack details

Provide a stack name

Stack name: iot-workshop
Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters
Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Console access
* Console Administrator Email
The user E-Mail to access the UI
Enter String

Cancel Previous Next

- On the “Configure stack options” page, leave all the defaults and click “Next”

CloudFormation > Stacks > Create stack

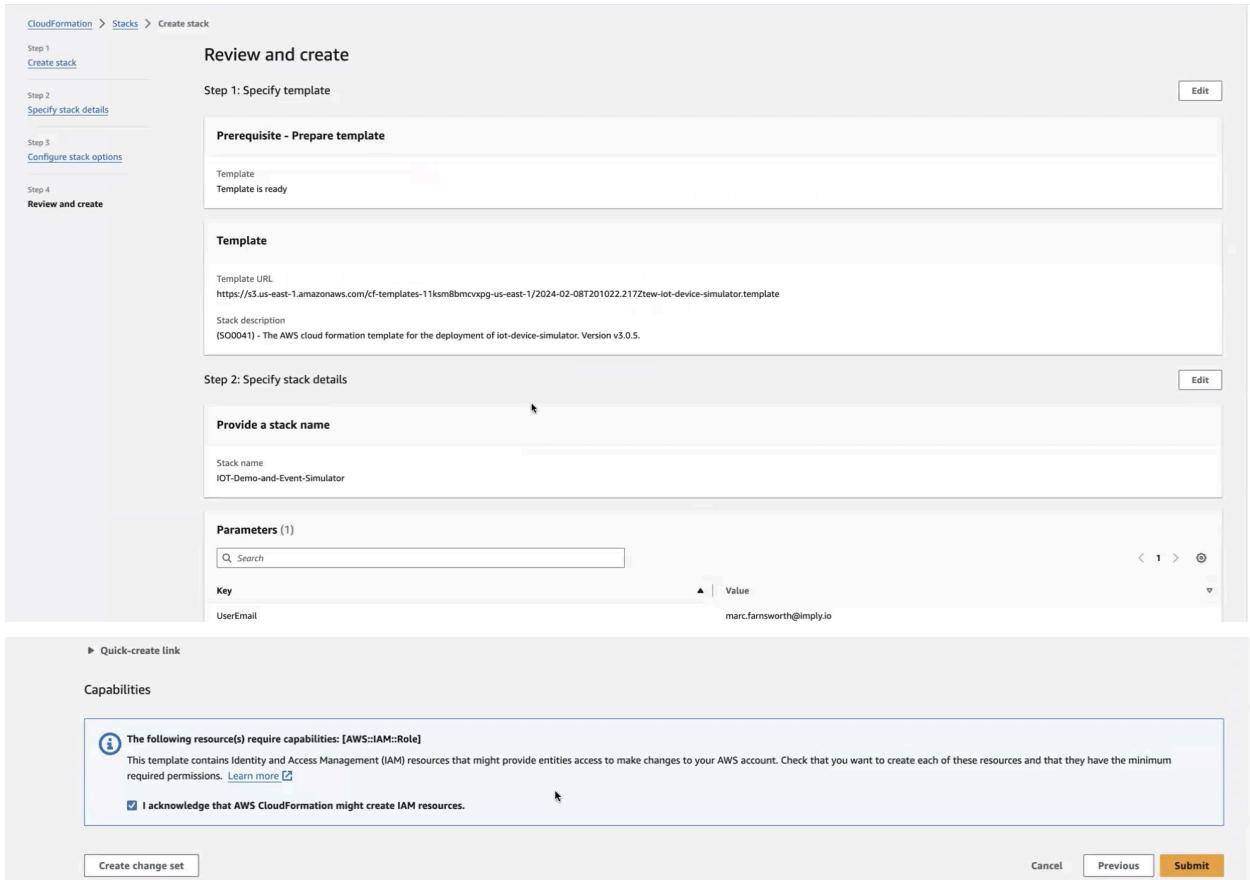
Configure stack options

Tags
You can specify tags (key-value pairs) to apply to resources in your stack. You can add up to 50 unique tags for each stack.
No tags associated with the stack.
Add new tag
You can add 50 more tag(s)

Permissions
IAM role - optional
Choose the IAM role for CloudFormation to use for all operations performed on the stack.
IAM role name: Sample-role-name
Remove

Stack failure options
Behavior on provisioning failure
Specify the roll back behavior for a stack failure. Learn more [?](#)
 Roll back all stack resources
Roll back the stack to the last known stable state.
 Preserve successfully provisioned resources
Preserves the state of successfully provisioned resources, while rolling back failed resources to the last known stable state. Resources without a last known stable state will be deleted upon the next stack operation.
Delete newly created resources during a rollback
Specify whether resources that were created during a failed operation should be deleted regardless of their deletion policy. Learn more [?](#)
 Use deletion policy
Retains or deletes created resources according to their attached deletion policy.
 Delete all newly created resources
Deletes created resources during a rollback regardless of their attached deletion policy.

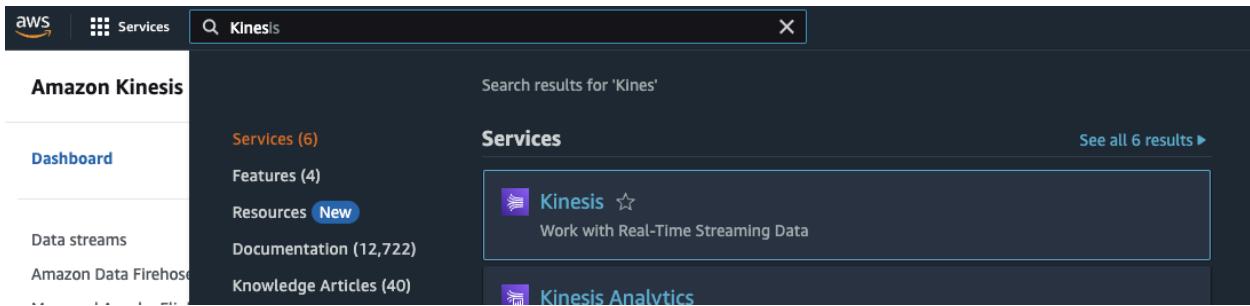
- On the next page, “Review and create”, check the box “I acknowledge that AWS CloudFormation might create IAM resources.” at the bottom of the page, and click “Submit” to begin the stack creation



8. Creating the stack can take some some minutes to finish setup
9. Upon completion of the project, you will get an email from AWS inviting you to join your newly created simulator. The email will have a link should have a link to the environment, your username, and temporary password (which you will need to change)

Step 2 - Setup of the Kinesis stream

1. Search for “Kinesis” on the AWS search bar, and click “Kinesis”



2. Click “Create data stream”

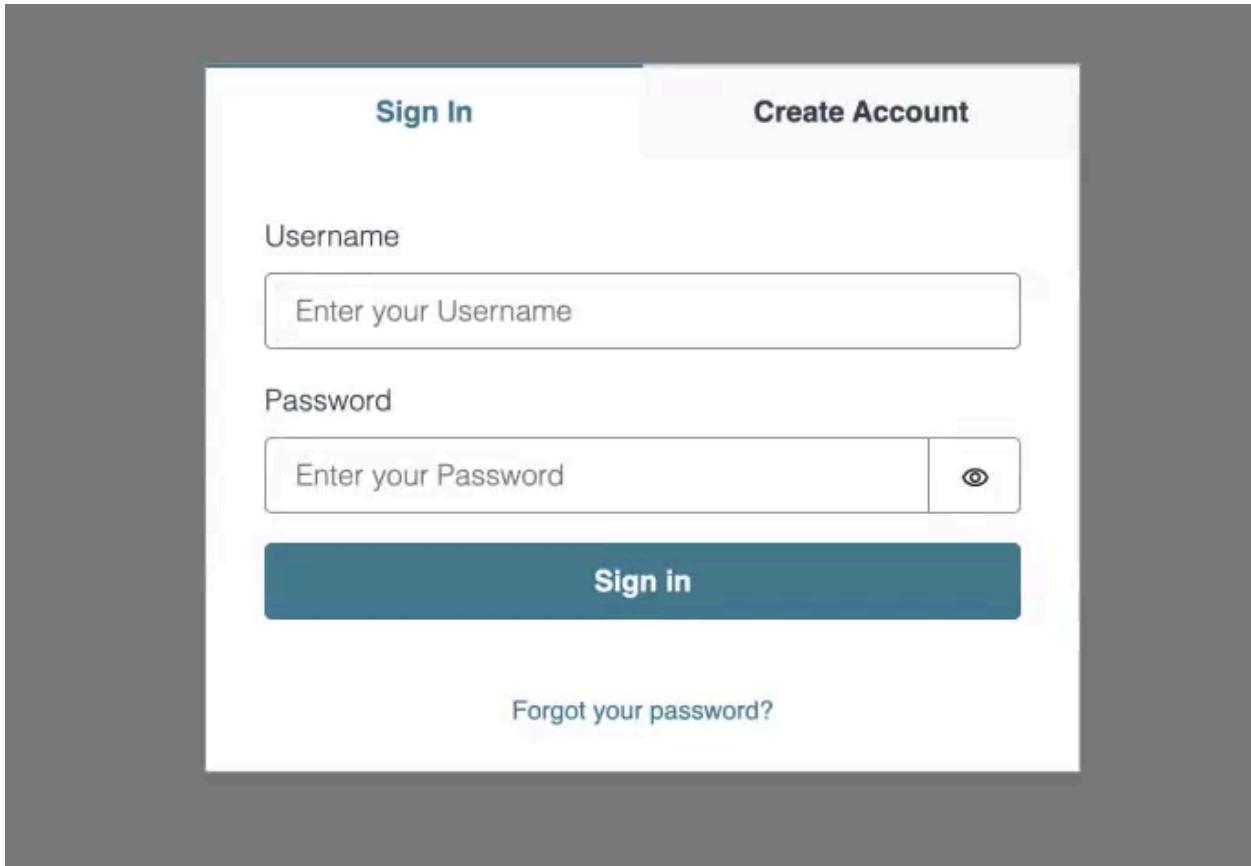
The screenshot shows the Amazon Kinesis Dashboard. On the left sidebar, under the "Data streams" section, there are links for "Amazon Data Firehose" (New) and "Managed Apache Flink" (New). Under the "Resources" section, there are links for "CloudFormation templates" and "AWS Glue Schema Registry". The main content area is titled "Amazon Kinesis Info" and describes the service's purpose. It shows a summary of "Data Streams": "Total data streams 5" and a "Create data stream" button. To the right, there are sections for "Amazon Data Firehose" and "Managed Apache Flink".

3. Provide a “Data stream name”, leave all other defaults, and click “Create data stream”

The screenshot shows the "Create data stream" configuration page. At the top, the breadcrumb navigation shows "Amazon Kinesis > Data streams > Create data stream". The main section is titled "Data stream configuration" and contains a "Data stream name" field with the value "iot-workshop". Below it, a note says "Acceptable characters are uppercase and lowercase letters, numbers, underscores, hyphens and periods." The next section is titled "Data stream capacity" and includes a "Capacity mode" section. It shows two options: "On-demand" (selected) and "Provisioned". The "On-demand" option is described as "Use this mode when your data stream's throughput requirements are unpredictable and variable. With on-demand mode, your data stream's capacity scales automatically." The "Provisioned" option is described as "Use provisioned mode when you can reliably estimate throughput requirements of your data stream. With provisioned mode, your data stream's capacity is fixed." At the bottom, there is a note about "Total data stream capacity" and a link to "View details".

Step 3 - Create your data simulator

1. Login to your IoT Device Simulator with the username and temporary password that was emailed to you, and it will prompt you to change your password



2. This should be what your landing page looks like, next click on “Device Types” at the top of the page

A screenshot of the IoT Device Simulator's landing page. The top navigation bar includes links for "IoT Device Simulator", "Simulations", and "Device Types". The main content area is titled "Simulations" and shows "Home > Simulations". It displays a message "There are no simulations to display" and a sub-message "Create a device type to get started." Below this, it says "If a device type has already been created, create a simulation to begin simulating devices." At the bottom, there is a note "For help please see the solution's home page" with a link. On the right side of the page, there are buttons for "Add Simulation" and "Refresh", and status indicators for "Devices 0 running" and "Simulations 0 running".

3. Click “ + Add Device Type”

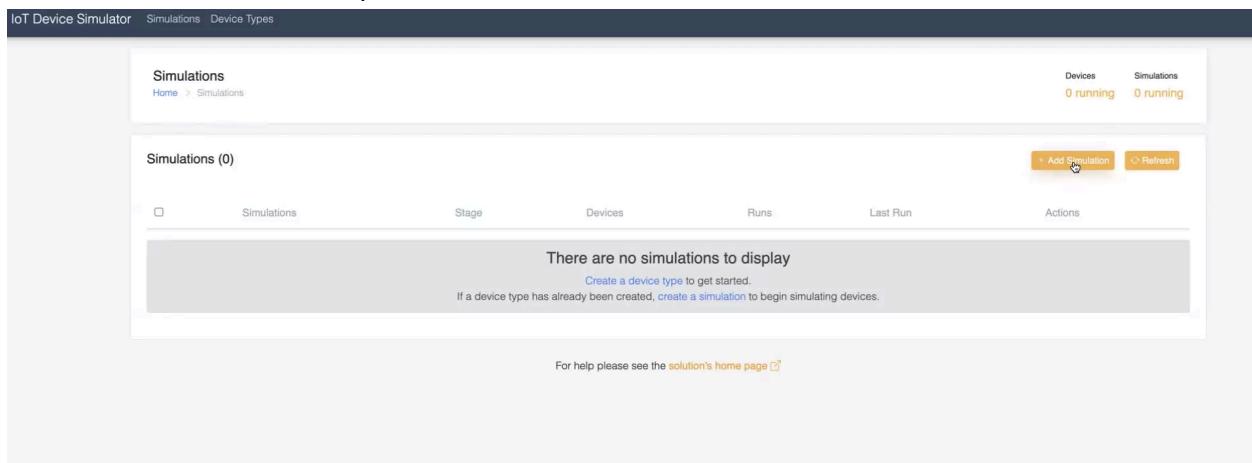
The screenshot shows the 'Device Types' page of the IoT Device Simulator. The top navigation bar includes 'IoT Device Simulator', 'Simulations', and 'Device Types'. On the right side, there are buttons for 'Devices' (0 running) and 'Simulations' (0 running). The main content area is titled 'Device Types (0)' and displays a message: 'There are no device types to display' with a link 'Create a device type to get started'. At the bottom, there is a help link 'For help please see the solution's home page'.

4. Fill out the “Device type name” as “Auto”, fill out the “Topic” as “simulation/auto” click “Automotive Demo” then click “Save”

The screenshot shows the 'Create Device Type' page. The top navigation bar includes 'IoT Device Simulator', 'Simulations', and 'Device Types'. On the right side, there are buttons for 'Devices' (0 running) and 'Simulations' (0 running). The main content area is titled 'Device Type Definition' and shows fields for 'Device type name' (set to 'Auto') and 'Topic' (set to 'simulation/auto'). Below these, there is a section for 'Message payload' with a table of message attributes. The table has columns for 'Message attribute', 'Data type', 'Static value', and 'Actions'. The attributes listed are: VIN (id), tripId (id), brake (float), steeringWheelAngle (float), torqueAtTransmission (float), engineSpeed (float), vehicleSpeed (float), and acceleration (float). Each row has 'View' and 'Delete' buttons. At the bottom, there is a 'Save' button.

The screenshot then shows the 'Device Types' page again, but this time it lists a single device type named 'Auto' with a topic of 'simulation/auto'. The 'Edit' and 'Delete' buttons are visible next to the entry.

5. Click “Simulations” at the top left corner, click “+ Add Simulation”



6. On the “Create a Simulation” page, name the simulation, select the “Automotive Demo” from the simulation type, select “Auto” from the device type, set the number of devices to 10, and change the “Data transmission duration” to 1000 and then save
- To use a custom evAuto, you can import this json into the Device type instead.
[evDataUser.json](#)

IoT Device Simulator Simulations Device Types

Create Simulation
Home > Simulations > Create

Devices Simulations
0 running 0 running

Create A Simulation
Create a custom simulation to run

Simulation name: auto-sim

Simulation type: Automotive Demo

Select a device type: Auto Number of devices: 10

Data transmission interval: 1

Data transmission duration: 1000

Save Cancel

IoT Device Simulator Simulations Device Types

Simulations
Home > Simulations

Devices Simulations
0 running 0 running

Simulations (1)

<input type="checkbox"/>	Simulations	Stage	Devices	Runs	Last Run	Actions
<input checked="" type="checkbox"/>	auto-sim	sleeping	Info	0		View Delete

For help please see the [solution's home page](#)

7. Check the box on the left of the newly created simulation, and click “Start simulation(s)”

IoT Device Simulator Simulations Device Types

Simulations
Home > Simulations

Devices Simulations
0 running 0 running

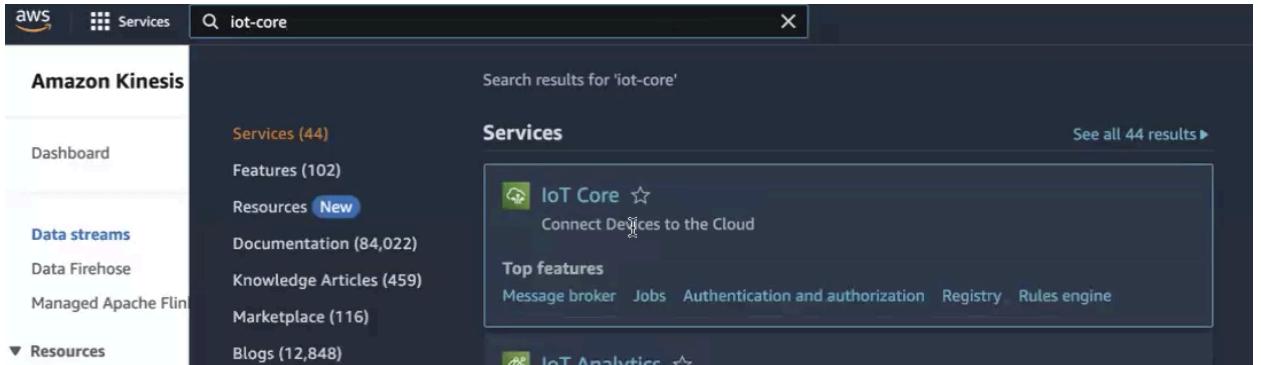
Simulations (1)

<input checked="" type="checkbox"/>	Simulations	Stage	Devices	Runs	Last Run	Actions
<input checked="" type="checkbox"/>	auto-sim	running	Info	0		View Delete

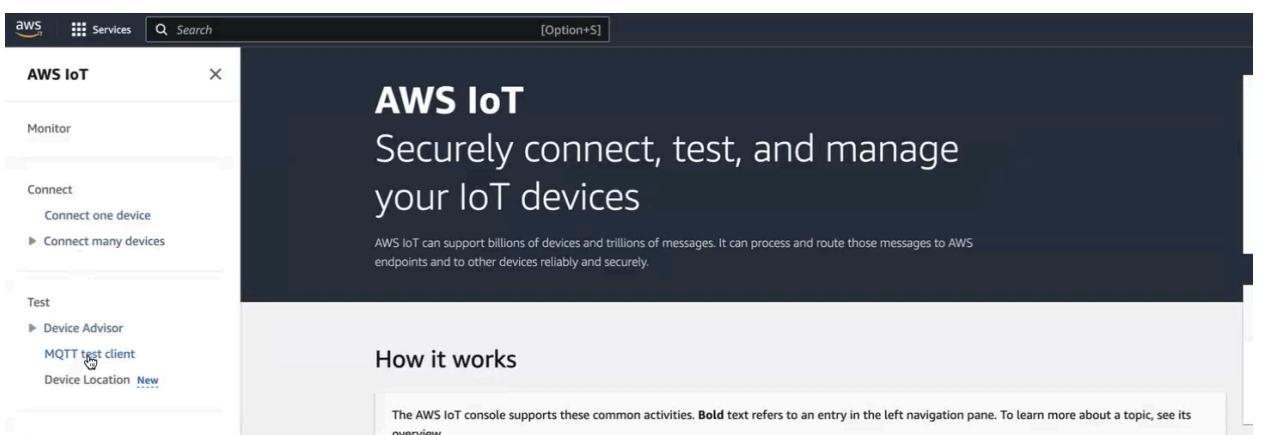
For help please see the [solution's home page](#)

Step 4 - Connect the simulator to AWS IoT Core, and stream to Kinesis

1. Search for “iot-core” and navigate there



2. On the left menu click on “MQTT test client”



3. Test a subscription to the topic, enter “simulation/auto” as the topic filter, and click “Subscribe”, you should now see events coming through

The screenshot shows the AWS IoT MQTT test client interface. On the left, a sidebar menu includes options like Monitor, Connect (Connect one device, Connect many devices), Test (Device Advisor, MQTT test client, Device Location), Manage (All devices, Greengrass devices, LPWAN devices, Software packages), Remote actions, Message routing, Retained messages, and Security.

The main content area is titled "MQTT test client" and shows a "Connection details" section with a note about updating connection details via Disconnect and Establish connection. Below this are two tabs: "Subscribe to a topic" (selected) and "Publish to a topic".

The "Topic filter" field contains "simulation/auto". A "Subscribe" button is visible. The "Subscriptions" tab shows no subscriptions, and the "Topic" tab shows a placeholder "Subscribe or select a topic to view incoming messages".

A modal window for the topic "simulation/auto" is open, showing a "Message payload" field containing the JSON message: { "message": "Hello from AWS IoT console" }. It also has an "Additional configuration" section and a "Publish" button.

The bottom section displays a log entry for the topic "simulation/auto" dated February 08, 2024, at 13:36:53 (UTC-0700). The log message is identical to the published message: { "timestamp": "2024-02-08 20:36:53.903000000", "trip.id": "VS9GkDrRpBhdayPfnz3C", "VIN": "JUGUJMLCXAT0JVVE", "brake": 0, "steeringWheelAngle": 0, "torqueAtTransmission": 46.4, "engineSpeed": 2952.74, "vehicleSpeed": 53.94, "acceleration": -0.3043, "parkingBrakeStatus": false, "brakePedalStatus": false, "transmissionGearPosition": "third", "gearLeverPosition": "drive", "odometer": 0.805, "ignitionStatus": "run", "fuelLevel": 99.97, "fuelConsumedSinceRestart": 0.013989, "oilTemp": 113.8, "location": { "latitude": 38.863508, "longitude": -77.042896 }, "_id": "K9wDy0" }.

- On the left menu, click on “Message routing”, click then “Create rule”, and provide a “Rule name” then click “Next”

The screenshot shows the AWS IoT console with the 'Message routing' section selected. On the left, a sidebar lists 'Monitor', 'Connect' (with 'Connect one device' and 'Connect many devices' options), and 'Test' (with 'Device Advisor' and 'MQTT test client'). The main content area has a dark header with the title 'Message routing' and the sub-header 'Route your device data to other AWS services'. Below this, a sub-header reads 'Configure how the AWS IoT rules engine processes and routes data from your devices to AWS and external services.' A call-to-action button 'Create rule' is visible. The URL in the browser bar is 'AWS IoT > Message routing > Rules > Create rule'. The main form is titled 'Specify rule properties' and includes a sub-section 'Rule properties'. It features a 'Rule name' input field containing 'autoSimRouting', a 'Rule description - optional' input field containing 'A description of your new rule', and a 'Tags - optional' section with a 'Add new tag' button. Navigation buttons 'Cancel' and 'Next' are at the bottom right.

AWS IoT

Monitor

Connect

Connect one device

Connect many devices

Test

Device Advisor

MQTT test client

AWS IoT

Message routing

Route your device data to other AWS services

Create a rule

Rules route data from your devices to AWS and other web services. Rules are evaluated and perform actions based on messages published by your devices.

Create rule

Pricing

AWS IoT > Message routing > Rules > Create rule

Step 1
Specify rule properties

Step 2
Configure SQL statement

Step 3
Attach rule actions

Step 4
Review and create

Specify rule properties Info

A rule resource contains a list of actions based on the MQTT topic stream.

Rule properties

Rule name

autoSimRouting

Enter an alphanumeric string that can also contain underscore (_) characters, but no spaces.

Rule description - *optional*

Enter a description to provide additional details about the rule to others.

A description of your new rule

▼ Tags - *optional*

No tags associated with the resource.

Add new tag

You can add up to 50 tags.

Cancel

Next

5. Configure the rule as “SELECT * FROM ‘simulation/auto’”, then click “Next”

The screenshot shows the AWS IoT Rules 'Create rule' wizard. The current step is 'Step 2: Configure SQL statement'. The top navigation bar shows 'AWS IoT > Message routing > Rules > Create rule'. The main title 'Configure SQL statement' has an 'Info' link. Below it, a note says 'Add a simplified SQL syntax to filter messages received on an MQTT topic and push the data elsewhere.' A sidebar on the left lists steps: Step 1 (Specify rule properties), Step 2 (Configure SQL statement, currently selected), Step 3 (Attach rule actions), and Step 4 (Review and create). The main area is titled 'SQL statement'. It includes a 'SQL version' dropdown set to '2016-03-23'. The 'SQL statement' text input field contains the query '1 SELECT * FROM \'simulation/auto\''. A status bar at the bottom indicates 'SQL Ln 1, Col 32'. At the bottom right are 'Cancel', 'Previous', and 'Next' buttons, with 'Next' being highlighted.

6. Configure for a “Kinesis stream” by selecting from the action dropdown, select the previously created stream “iot-workshop” from the “Stream name” dropdown, and add a “Partition key”,

Step 2
[Configure SQL statement](#)

Step 3
Attach rule actions

Step 4
Review and create

SQL statement

```
SELECT * FROM 'simulation/auto'
```

Rule 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. You can add up to 10 actions.

Action 1

Kinesis Stream
A message sent to a VPC will be metered with an additional action.
Stream name [Info](#)
IOT-demo [C](#) [View](#) [Create Kinesis Stream](#)

Partition key
The partition key is used to group data by shard in a stream. Try a generator function like \${newuuid()}.
\${VIN}

IAM role
Choose a role to grant AWS IoT access to your endpoint.
Choose an IAM role [C](#) [View](#) [Create new role](#)
AWS IoT will automatically create a policy with a prefix of "aws-iot-rule" under your IAM role selected.

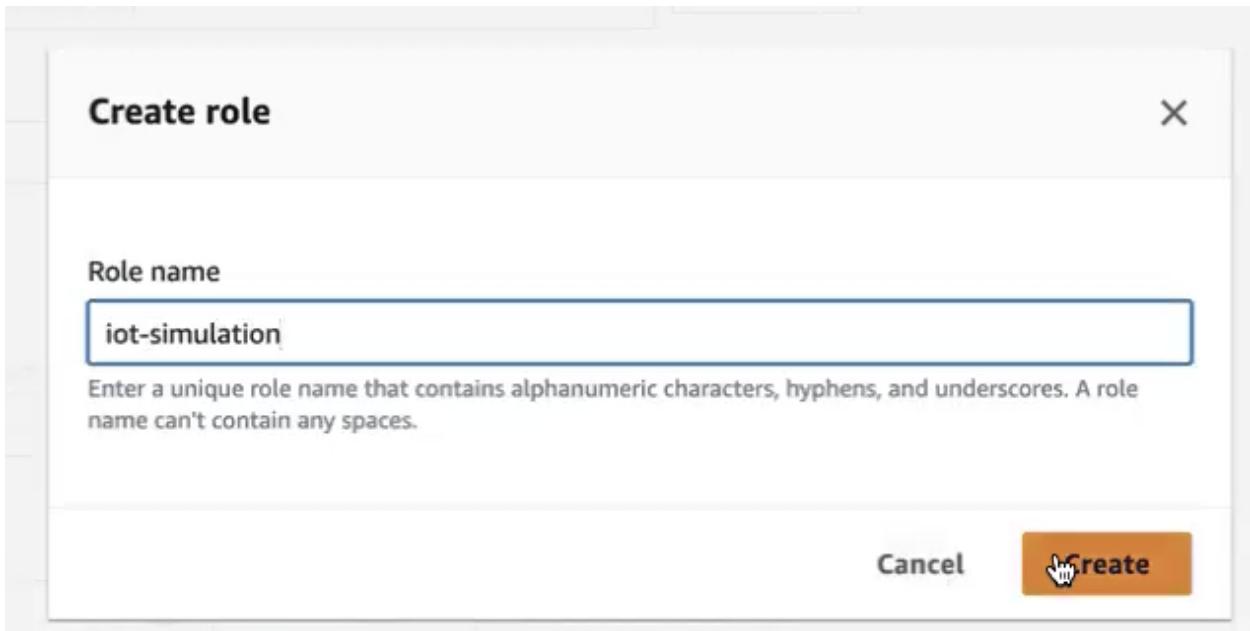
[Add rule action](#)

Error action - optional
You can optionally set an action that will be executed when something goes wrong with processing your rule. If two rule actions in the same rule fail, the error action receives one message that contains both errors.

[Add error action](#)

[Cancel](#) [Previous](#) **Next**

7. On this same page for the action we are working on, we will want to “Create new role”



8. For the “Error action” section, select “Republish to AWS IoT topic” from the drop down, and select the same IAM role and click “Next”

Error action - optional

You can optionally set an action that will be executed when something goes wrong with processing your rule. If two rule actions in the same rule fail, the error action receives one message that contains both errors.

▼ Republish to AWS IoT topic Remove

Topic [Info](#)
simulation/error

Quality of service
When subscribing to a topic, quality of service 0 is chosen by default.
 0 - The message is delivered at most once
 1 - The message is delivered at least once

MQTT 5 properties [Info](#)
Add MQTT 5 properties by choosing a property type and adding a value.

Add property
You can add 5 more properties.

User properties [Info](#)
Add user properties in the MQTT header.

Add user property

IAM role
Choose a role to grant AWS IoT access to your endpoint.
iot-simulation ▼ C View Create new role

AWS IoT will automatically create a policy with a prefix of "aws-iot-rule" under your IAM role selected.

Add error action

Cancel Previous Next

9. Next on the review page click “Create”

The screenshot shows the AWS IoT Rule creation process. The top navigation bar indicates "Successfully created role iot-simulation". The main content is divided into two steps:

- Step 2: SQL statement**: Shows the SQL query: "SELECT * FROM 'simulation/auto'".
- Step 3: Rule actions**: Contains two sections:
 - Actions**: Set up to send data to a Kinesis Stream named "IOT-demo" using the partition key "\${VIN}". The IAM role is "arn:aws:iam::169224085381:role/service-role/iot-simulation".
 - Error action**: Set up to republish messages to an AWS IoT topic "simulation/error" with Quality of Service 0.

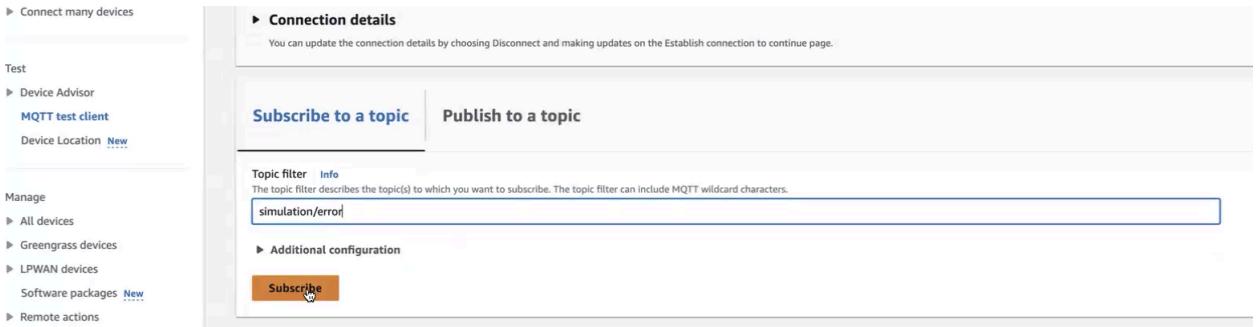
At the bottom right, there are "Cancel", "Previous", and "Create" buttons. The "Create" button is highlighted with a mouse cursor.

The screenshot shows the AWS IoT Rules list page. The left sidebar includes "AWS IoT", "Monitor", "Connect" (with "Connect one device" and "Connect many devices" options), "Test" (with "Device Advisor", "MQTT test client", and "Device Location" options), and a "New" button. The main content area shows the "Rules (1) Info" section with the following details:

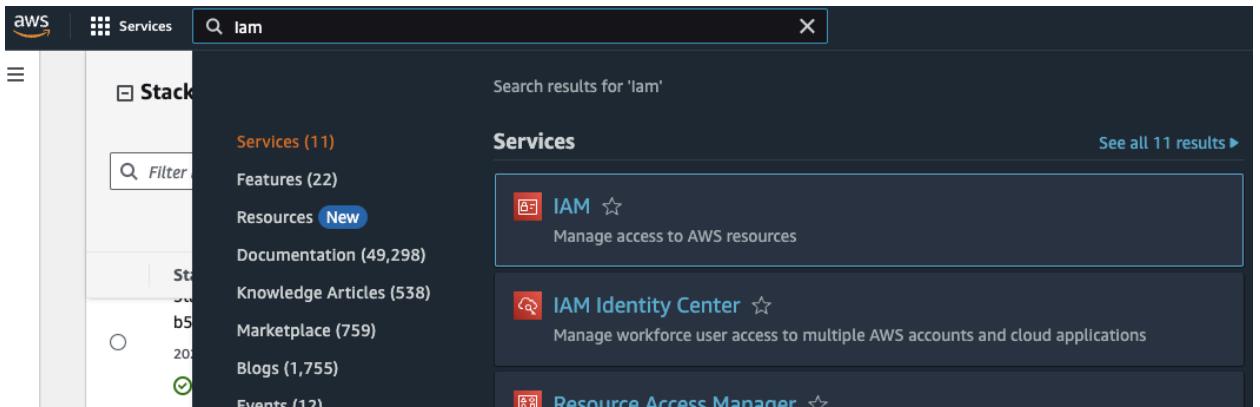
Name	Status	Rule topic	Created date
autoSimRouting	Active	simulation/auto	February 08, 2024, 13:40:31 (UTC-07:00)

At the top right of the rules list, there are buttons for "Activate", "Deactivate", "Edit", "Delete", and "Create rule".

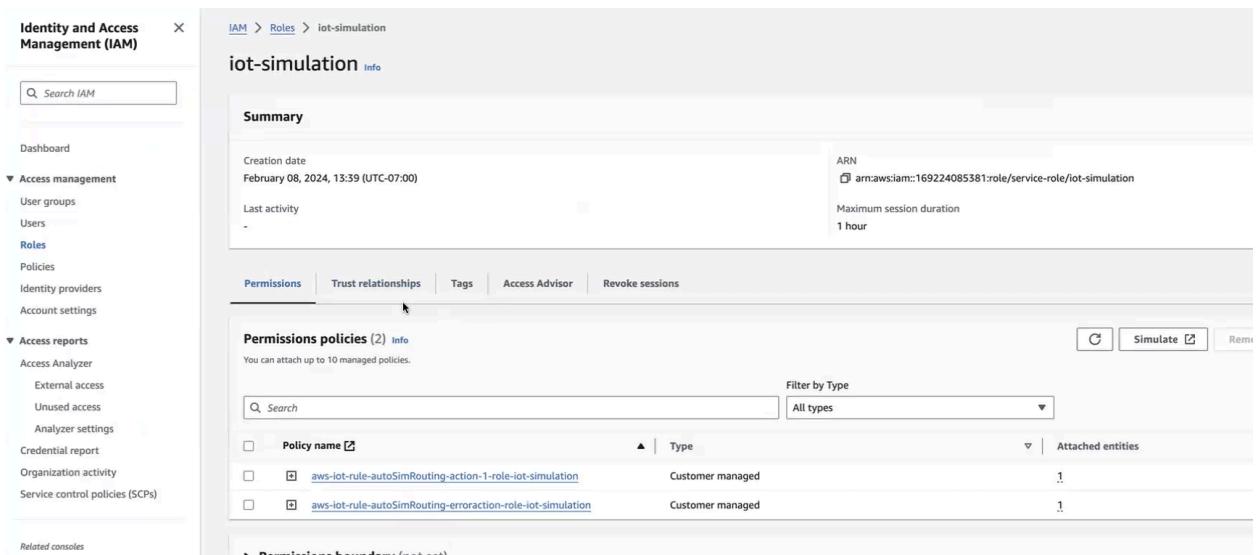
10. On the left menu click on “MQTT test client” and change the topic to “simulation/error” and click “subscribe”, until we add permissions in IAM our events will be going to the error topic



11. In another window, go to the IAM dashboard, and click “Roles”



12. Find the newly created “iot-simulation” role, click the “Add permissions” button under the Permission policies section and click “attach policies”



13. Add the “AmazonKinesisFullAccess” policy

The screenshot shows the 'Add permissions' dialog in the AWS IAM console. The search bar at the top contains 'kines'. Below it, a table lists various AWS managed policies. One row, 'AmazonKinesisFullAccess', has a checked checkbox. The table includes columns for 'Policy name', 'Type', and 'Description'. At the bottom right of the dialog are 'Cancel' and 'Add permissions' buttons.

14. Navigate back to the MQTT test client, error events should no longer be streaming in

The screenshot shows the 'MQTT test client' section of the AWS IoT console. It displays two green success notifications: 'Successfully created role iot-simulation' and 'Successfully created rule autoSimRouting'. Below these, there are tabs for 'Subscribe to a topic' and 'Publish to a topic'. Under 'Subscribe to a topic', a topic filter 'simulation/error' is set, and a 'Subscribe' button is visible. On the right, a 'Subscriptions' panel shows a single entry for 'simulation/error' with a message payload containing a JSON object: {"message": "Hello from AWS IoT console"}. At the bottom right, a timestamp indicates the data is from February 08, 2024, 13:41:41 (UTC-0700).

15. Navigate back to Kinesis, click on “Data streams”, and click the “iot-workshop” stream

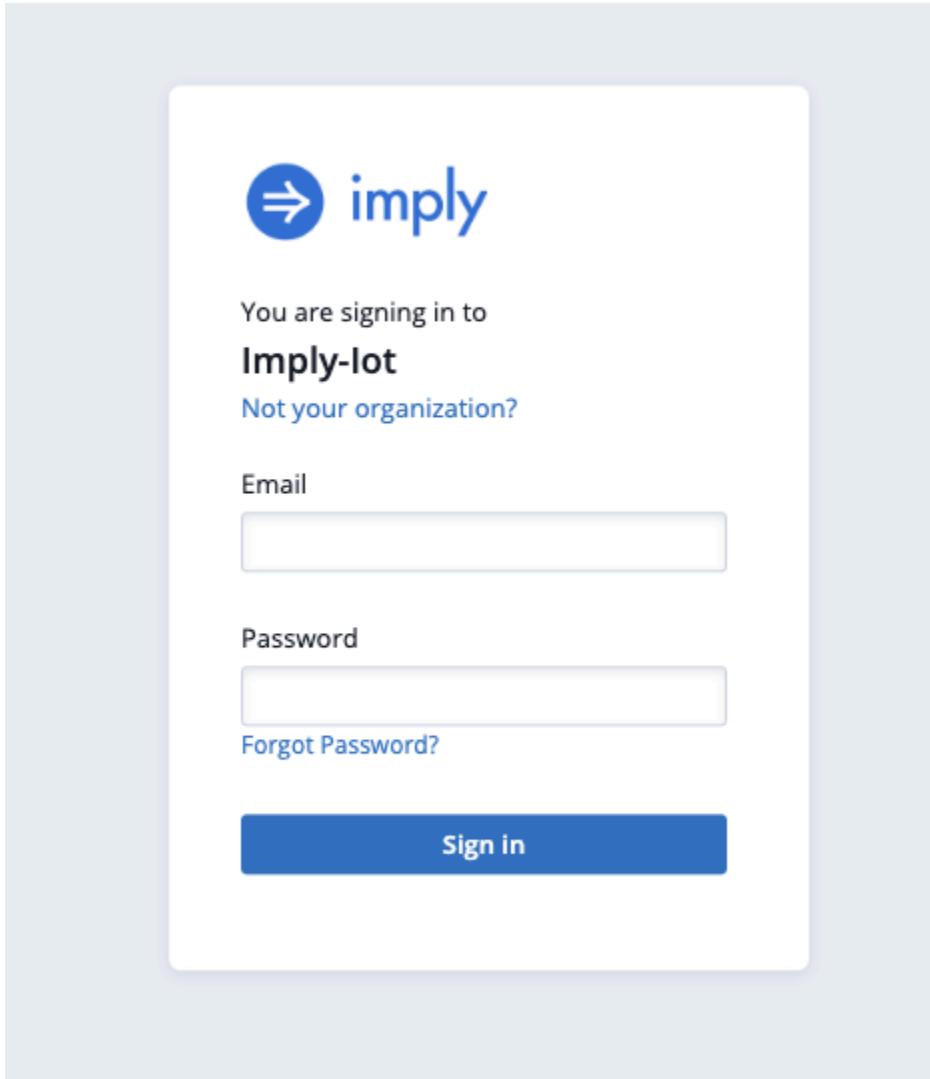
The screenshot shows the Amazon Kinesis service dashboard. On the left, there's a sidebar with 'Amazon Kinesis' at the top, followed by 'Dashboard', 'Data streams' (which is highlighted with a mouse cursor), 'Data Firehose', and 'Managed Apache Flink'. Below that is a 'Resources' section with 'CloudFormation templates' and 'AWS Glue Schema Registry'. The main content area is titled 'Amazon Kinesis' and 'Info'. It says 'Amazon Kinesis makes it easy to collect, process, and analyze data streams in real time, so you can...'. A large box titled 'Data Streams' contains the text 'Total data streams' and the number '5'. Below this is a 'Create data stream' button.

16. Click on “Data viewer” tab in the middle of the screen, under “Shard” select the first one, and then on the right side click “Next record” to validate events are now on the stream

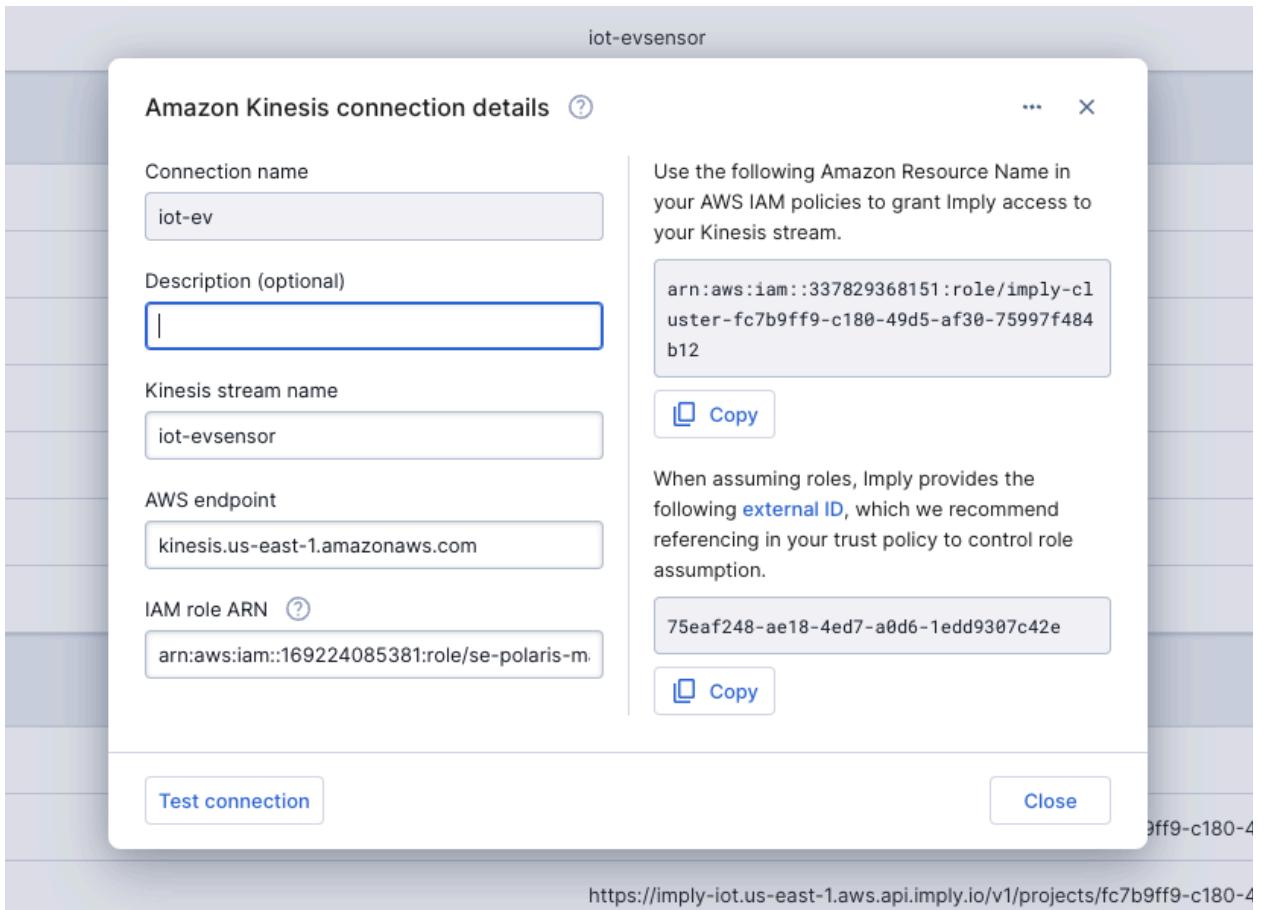
The screenshot shows the 'IOT-demo' data stream details page. At the top, it has a 'Data stream summary' section with fields for Status (Active), Capacity mode (On-demand), ARN (arn:aws:kinesis:us-east-1:169224085381:stream/IOT-demo), and Creation time (February 08, 2024 at 13:22 MST). Below this is a navigation bar with tabs: Applications, Monitoring, Configuration, Data viewer (which is selected and highlighted in blue), Enhanced fan-out (0), Data stream sharing - new, and EventBridge Pipes. Under the 'Data viewer' tab, there's a 'Shard' dropdown set to 'shardid-000000000000'. To the right, there's a 'Starting position' dropdown set to 'Latest' and a 'Get records' button. The main area is titled 'Records (8)' and lists 8 records. Each record includes a 'Partition key' (e.g., FUOCRFTAFJIP...), 'Data' (e.g., {"timestamp": "2024-02-08 20:42:44.771000000", "trip_id": "AwlW4w6..."}, {"timestamp": "2024-02-08 20:42:44.849000000", "trip_id": "HHUJ-pCm..."}), 'Approximate arrival timestamp' (e.g., February 08, 2024 at 13:42:44 MST), and 'Sequence number' (e.g., 49649086454270979036686278508245359363826621369383124994, 4964908645427097903668627850824568289646235958557831170).

Step 5 - Connect Imply Polaris to Kinesis, and Create a Table

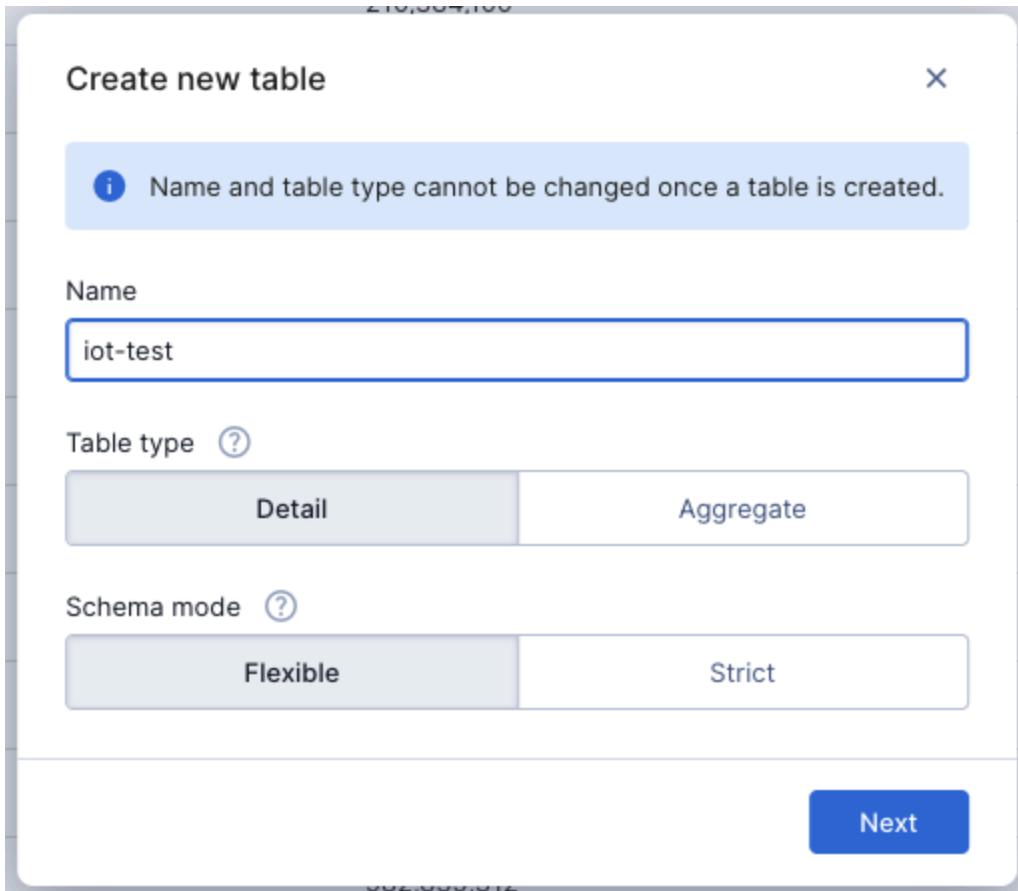
1. Sign into your Imply account



2. Click on “Sources” on the left nav, then at the top right click “Create source”. For a comprehensive guide, here are the docs
<https://docs.imply.io/polaris/ingestion-guide-kinesis>



3. Open IAM in AWS and add ARN from the polaris provided ARN to the relevant kinesis role with trust policies
4. Create a table in Imply Polaris by clicking on the “Tables” tab and click “Create table” at the top right



- Click “Load data” and “Insert data”, select “Amazon Kinesis”, and select the “auto-simulation” stream then click “Next”

Name	Stream name	Active jobs
auto-simulation	IOT-demo	1

6. Polaris will auto identify they schema, click “Continue”

String timestamp	String trip_id	String VIN	Long brake	Long steeringWheelAngle	Double torqueAtTransmission	Double engineSpeed	Double vehicleSpeed	Double acceleration	String parkingBrakeStatus	String brakePedalStatus	
2024-02-08 20:40:37.717...	mtSibJ4WYWosLZ0...	JNRZVHNPD2XTX4...	0	0	345.7	3597.51	87.61	-0.5446	false	false	
2024-02-08 20:40:38.033...	VSG9GkDrBhRdayPf...	JU6UJMLCXATOJV8E	0	-39.1	2588.59	31.35	-0.544	false	false	false	
2024-02-08 20:40:38.052...	AuKIBgaMW8w7ZWg...	VVQ4CZQHE8V35H8...	0	0	261.4	3161.95	77.24	0.0833	false	false	
2024-02-08 20:40:38.548...	Pugbr9HY05dTQgQ...	KDWBC39EIOR2HQ...	0	0	-16.9	2323.89	28.37	-0.013	false	false	
2024-02-08 20:40:38.717...	mtSibJ4WYWosLZ0...	JNRZVHNPD2XTX4...	0	0	346.3	3579.82	87.22	-0.4319	false	false	
2024-02-08 20:40:39.052...	AuKIBgaMW8w7ZWg...	VVQ4CZQHE8V35H8...	0	0	261.3	3164.56	77.3	0.0692	false	false	
2024-02-08 20:40:39.033...	VSG9GkDrBhRdayPf...	JU6UJMLCXATOJV8E	0	0	56.6	2634.62	32.97	-0.5049	false	false	
2024-02-08 20:40:39.548...	Pugbr9HY05dTQgQ...	KDWBC39EIOR2HQ...	0	0	-16.9	2322.89	28.35	-0.013	false	false	
2024-02-08 20:40:39.718...	mtSibJ4WYWosLZ0...	JNRZVHNPD2XTX4...	0	0	346.8	3565.55	86.91	-0.3485	false	false	
2024-02-08 20:40:40.034...	VSG9GkDrBhRdayPf...	JU6UJMLCXATOJV8E	0	0	52.6	2764.41	34.51	1.6139	false	false	
2024-02-08 20:40:40.068...	AuKIBgaMW8w7ZWg...	VVQ4CZQHE8V35H8...	0	0	261.2	3168.76	77.35	0.0585	false	false	
2024-02-08 20:40:40.549...	Pugbr9HY05dTQgQ...	KDWBC39EIOR2HQ...	0	0	-16.9	2321.93	28.34	-0.0182	false	false	
2024-02-08 20:40:40.718...	mtSibJ4WYWosLZ0...	JNRZVHNPD2XTX4...	0	0	347.2	3554	86.66	-0.282	false	false	
2024-02-08 20:40:41.034...	VSG9GkDrBhRdayPf...	JU6UJMLCXATOJV8E	0	0	48.7	2886.37	35.95	1.5375	false	false	
2024-02-08 20:40:41.069...	AuKIBgaMW8w7ZWg...	VVQ4CZQHE8V35H8...	0	0	261.2	3168.62	77.39	0.0494	false	false	
2024-02-08 20:40:41.549...	Pugbr9HY05dTQgQ...	KDWBC39EIOR2HQ...	0	0	-16.9	2321.03	28.33	-0.0113	false	false	
2024-02-08 20:40:41.719...	mtSibJ4WYWosLZ0...	JNRZVHNPD2XTX4...	0	0	347.5	3544.63	86.45	-0.2286	false	false	
2024-02-08 20:40:42.034...	VSG9GkDrBhRdayPf...	JU6UJMLCXATOJV8E	0	0	85.6	3037.74	38.2	1.4439	false	false	
2024-02-08 20:40:42.068...	AuKIBgaMW8w7ZWg...	VVQ4CZQHE8V35H8...	0	0	292.6	3179.41	77.86	0.0417	false	false	
2024-02-08 20:40:42.550...	Pugbr9HY05dTQgQ...	KDWBC39EIOR2HQ...	0	0	-43.8	2295.6	2772	-0.0108	false	false	
2024-02-08 20:40:42.719...	mtSibJ4WYWosLZ0...	JNRZVHNPD2XTX4...	0	0	410.7	3555.5	87.14	0.2654	false	false	
2024-02-08 20:40:43.033...	VSG9GkDrBhRdayPf...	JU6UJMLCXATOJV8E	0	0	0	0	0	0	2.2481	false	false
2024-02-08 20:40:43.069...	AuKIBgaMW8w7ZWg...	VVQ4CZQHE8V35H8...	0	0	0	0	0	0	0.469	false	false

7. You can customize the table further, but for now in this test click “Start Ingestion”

String timestamp	Auto trip_id	Auto VIN	Auto brake	Auto steeringWheelAngle	Auto torqueAtTransmission	Auto engineSpeed	Auto vehicleSpeed	Auto acceleration	Auto parkingBrakeStatus	Auto brakePedalStatus	Auto transmissionGearPosition	Auto gearLevel	
2024-02-08T20:40:37.717Z	mtSibJ4WYWosLZ0...	JNRZVHNPD2XTX4...	0	0	345.7	3597.51	87.61	-0.5446	false	false	fourth	drive	
2024-02-08T20:40:38.030Z	VSG9GkDrBhRdayPf...	JU6UJMLCXATOJV8E	0	-39.1	2588.59	31.35	-0.544	false	false	false	second	drive	
2024-02-08T20:40:38.050Z	AuKIBgaMW8w7ZWg...	VVQ4CZQHE8V35H8...	0	0	261.4	3161.95	77.24	0.0833	false	false	fourth	drive	
2024-02-08T20:40:38.548Z	Pugbr9HY05dTQgQ...	KDWBC39EIOR2HQ...	0	0	-16.9	2323.89	28.37	-0.013	false	false	second	drive	
2024-02-08T20:40:38.717Z	mtSibJ4WYWosLZ0...	JNRZVHNPD2XTX4...	0	0	346.3	3579.82	87.22	-0.4319	false	false	fourth	drive	
2024-02-08T20:40:39.050Z	AuKIBgaMW8w7ZWg...	VVQ4CZQHE8V35H8...	0	0	261.3	3164.56	77.3	0.0692	false	false	fourth	drive	
2024-02-08T20:40:39.033Z	VSG9GkDrBhRdayPf...	JU6UJMLCXATOJV8E	0	0	56.6	2634.62	32.97	-0.5049	false	false	second	drive	
2024-02-08T20:40:39.548Z	Pugbr9HY05dTQgQ...	KDWBC39EIOR2HQ...	0	0	-16.9	2322.89	28.35	-0.013	false	false	second	drive	
2024-02-08T20:40:39.717Z	mtSibJ4WYWosLZ0...	JNRZVHNPD2XTX4...	0	0	346.8	3565.55	86.91	-0.3485	false	false	fourth	drive	
2024-02-08T20:40:40.034Z	VSG9GkDrBhRdayPf...	JU6UJMLCXATOJV8E	0	0	52.6	2764.41	34.51	1.6139	false	false	second	drive	
2024-02-08T20:40:40.068Z	AuKIBgaMW8w7ZWg...	VVQ4CZQHE8V35H8...	0	0	261.2	3168.76	77.35	0.0585	false	false	fourth	drive	
2024-02-08T20:40:40.548Z	Pugbr9HY05dTQgQ...	KDWBC39EIOR2HQ...	0	0	-16.9	2321.93	28.34	-0.0182	false	false	second	drive	
2024-02-08T20:40:40.717Z	mtSibJ4WYWosLZ0...	JNRZVHNPD2XTX4...	0	0	347.2	3554	86.66	-0.282	false	false	fourth	drive	
2024-02-08T20:40:41.034Z	VSG9GkDrBhRdayPf...	JU6UJMLCXATOJV8E	0	0	48.7	2886.37	35.95	1.5375	false	false	second	drive	
2024-02-08T20:40:41.069Z	AuKIBgaMW8w7ZWg...	VVQ4CZQHE8V35H8...	0	0	261.2	3168.62	77.39	0.0494	false	false	fourth	drive	
2024-02-08T20:40:41.548Z	Pugbr9HY05dTQgQ...	KDWBC39EIOR2HQ...	0	0	-43.8	2295.6	2772	-0.0108	false	false	fourth	drive	
2024-02-08T20:40:41.719Z	mtSibJ4WYWosLZ0...	JNRZVHNPD2XTX4...	0	0	410.7	3555.5	87.14	0.2654	false	false	fourth	drive	
2024-02-08T20:40:43.033Z	VSG9GkDrBhRdayPf...	JU6UJMLCXATOJV8E	0	0	0	0	0	0	2.2481	false	false	drive	drive
2024-02-08T20:40:43.069Z	AuKIBgaMW8w7ZWg...	VVQ4CZQHE8V35H8...	0	0	0	0	0	0	0.469	false	false	drive	drive

8. Your table is now created, congratulations!!! You have created an end-to-end data application pipeline. You can now query your IoT event data from the UI or via API.

The screenshot shows the imply-iot interface with the 'auto-sim' table selected. The table has 19 columns and 23,788 rows. The columns include: _time, vehicleSpeed, trip_id, fuelLevel, brakePedalStatus, odometer, id_, fuelConsumedSinceRest, and several other auto-related properties. A 'Visualize your data' panel is open, showing a preview of the data.

9. Lets query the table:

- From the home menu, on the left side click on “SQL”, click the newly created table and “Run” the query.

The screenshot shows the imply-iot SQL interface. The left sidebar shows the navigation menu with 'SQL' selected. In the main area, the 'druid' database is selected, and the 'iot-ev' table is shown. A dropdown menu is open over the 'iot-ev' table, displaying the following SQL code:

```

1 SELECT
2   "_time",
3   "fault_properties",
4   "charging_properties",
5   "torque_achieved",
6   "fleet",
7   "subtype",
8   "event_timing",
9   "type",
10  "id_",
11  "car_ownership_id",
12  "seq",
13  "temperature_IGBT_A",
14  "motor_temperature",
15  "torque_available_motoring",
16  "temperature_IGBT_C",
17  "temperature_IGBT_B",
18  "VIN",
19  "updated",
20  "tenant".

```

Below this, there are several collapsed query options:

- SELECT ...columns... FROM iot-ev
- SELECT * FROM iot-ev
- SELECT COUNT(*) AS "Count" FROM iot-ev
- SELECT MIN(_time), MAX(_time) FROM iot-ev
- Copy: "iot-ev"

- Here are some example queries you can test in the SQL UI

Unset

```

SELECT
    TIME_FLOOR(CAST(t."__time" AS TIMESTAMP), 'PT1H', NULL, 'Etc/UTC') AS
    "__time",
    "tenant",
    (COUNT(*)) AS "count"
FROM "iot-ev" AS t
WHERE ((0 <= CAST(t."motor_temperature" AS DOUBLE) AND
CAST(t."motor_temperature" AS DOUBLE) < 50)) IS NOT TRUE
GROUP BY 1,2

SELECT
    TIME_FLOOR(CAST(t."__time" AS TIMESTAMP), 'PT1H', NULL, 'Etc/UTC') AS
    "__time",
    CAST(t."fleet" AS VARCHAR) AS "fleet",
    (COUNT(*)) AS "count"
FROM "iot-ev" AS t
WHERE CAST(t."fleet" AS VARCHAR) IN ('Fleet Van', 'Passenger Car', 'Fleet
Truck', 'Passenger Truck', 'Fleet Car')
GROUP BY 1, 2

```

The screenshot shows the Superset interface with the following details:

- Left Sidebar:** Home, Collections (with 'iot-ev' selected), DATA (Tables, Sources, Jobs), ANALYTICS (Data cubes, Dashboards, Alerts, Reports, Embedding), MONITORING (User queries, Streaming, Detailed metrics, Server).
- SQL Tab:** Shows the two queries listed above.
- Results Tab:** Shows the output of the first query (Tab 1). The table has columns: __time, tenant, # count (COUNT(*)). The data is as follows:

__time	tenant	# count (COUNT(*))
2024-03-27T15:00:00.000Z	rivian	17,224
2024-03-27T16:00:00.000Z	rivian	24,017
2024-03-27T17:00:00.000Z	rivian	23,901
2024-03-27T18:00:00.000Z	rivian	23,990
2024-03-27T19:00:00.000Z	rivian	23,950
2024-03-27T20:00:00.000Z	rivian	24,025
2024-03-27T21:00:00.000Z	rivian	23,930

Step 6 - Real time Analytics & Dashboarding

1. Creating a data cube can be done in just a couple short steps:
 - a. On the left menu, click “Data Cubes”, then at the top right “Create data cube”

The screenshot shows the Imply Data Platform interface. On the left, there is a navigation sidebar with various sections like Home, Collections, DATA (Tables, Sources), Jobs, SQL, ANALYTICS (Data cubes, Dashboards, Alerts, Reports, Embedding), and MONITORING (User queries, Streaming, Detailed metrics, Server). The 'Data cubes' section is currently selected. The main area displays a table of existing data cubes, each with columns for Name, Table, Data assets, Edited date, and a star/rate icon. A modal window titled 'Create new data cube' is open over the table. It has a dropdown labeled 'Source' set to 'From table', a search input field containing 'iot-ev', and a button 'Next: Create data cube'.

Name	Table	Data assets	Edited	...
IoT Ev - Old	iot-ev	25 dimensions, 11 measures	Mar 27, 2024, 4:51 PM	...
IoT Ev	iot-ev	24 dimensions, 12 measures	Mar 27, 2024, 4:50 PM	...
IoT Webinar	iot-webinar	11 dimensions, 11 measures	Mar 27, 2024, 3:55 AM	...
Auto Sim	auto-sim	11 dimensions, 11 measures	Mar 25, 2024, 3:50 AM	...
Vehicle Data	vehicle-data	11 dimensions, 11 measures	Mar 20, 2024, 10:49 AM	...
Tillmanfiber Demo	tillmanfiber-demo	11 dimensions, 11 measures	Mar 1, 2024, 9:10 AM	...
Distribitech 2024 Demo	distribitech-2024-demo	11 dimensions, 11 measures	Feb 25, 2024, 4:55 PM	...
Tillmanfiber	tillmanfiber	11 dimensions, 11 measures	Feb 9, 2024, 8:56 AM	...
Imply IoT EON	imply-iot-eon	11 dimensions, 11 measures	Jan 19, 2024, 4:09 AM	...
IoT Table 01	iot-table-01	27 dimensions, 13 measures	Dec 8, 2023, 6:40 AM	...
Emirates Demo	emirates-demo	26 dimensions, 14 measures	Nov 2, 2023, 3:54 AM	...
Dtest	dtest	9 dimensions, 6 measures	Oct 27, 2023, 12:52 PM	...
Noah Test	NoahTest	7 dimensions, 1 measure	Oct 26, 2023, 4:13 PM	...
Imply IoT	imply-iot	27 dimensions, 14 measures	Oct 25, 2023, 4:46 AM	...
Imply IoT	imply-iot	27 dimensions, 14 measures	Oct 25, 2023, 3:17 AM	...
Plants Streams	plants-streams	15 dimensions, 18 measures	Oct 17, 2023, 8:20 AM	...
Imply IoT Ann	imply-iot-ann	14 dimensions, 15 measures	Oct 8, 2023, 10:18 AM	...

- b. From the drop down select your new table we created “iot-test”, and click “Next: Create data cube”. Leave the “Auto-fill dimensions and measures” checked
- c. Now let's modify the **status** dimension and replace the default expression with this new one, it will help to simulate a failure status only when **fault_type = 'fault_6'**:

```
CASE WHEN JSON_VALUE("t"."fault_properties", '$.fault_type') in ('fault_6', 'fault_6') THEN 'Failure' else 'OK Status' END
```

Edit dimension

Name	Status
Description	
<input type="radio"/> Basic <input type="radio"/> Custom	
Formula	<pre>CASE WHEN JSON_VALUE("\$.fault_properties", '\$.fault_type') in ('fault_6', ' fault_6') THEN 'Failure' else 'OK Status' END</pre>
Documentation	
Type	String
Add URL	
<input type="button" value="Cancel"/> <input type="button" value="Save"/>	

- d. Now let's modify the **fault_type** dimension and replace the default expression with this new one in order to decode the fault type descriptions :

```
CASE WHEN JSON_VALUE("t"."fault_properties",  
'$fault_type') = 'fault_1' THEN 'charging level 1' WHEN  
JSON_VALUE("t"."fault_properties", '$fault_type') = 'fault_2'  
THEN 'charging level 2' WHEN JSON_VALUE("t"."fault_properties",  
'$fault_type') = 'fault_3' THEN 'fast charging' WHEN  
JSON_VALUE("t"."fault_properties", '$fault_type') = 'fault_4'  
THEN 'low battery' WHEN JSON_VALUE("t"."fault_properties",  
'$fault_type') = 'fault_5' THEN 'charging issue' WHEN  
JSON_VALUE("t"."fault_properties", '$fault_type') = 'fault_6'  
THEN 'engine overheating' END
```

Edit dimension

Name	Fault Type
Description	
<input type="radio"/> Basic <input type="radio"/> Custom	
Formula	<pre>CASE WHEN JSON_VALUE("\$.fault_properties", '\$fault_type') = 'fault_1' THEN 'charging level 1' WHEN JSON_VALUE("t"."fault_properties", '\$fault_type') = 'fault_2' THEN 'charging level 2' WHEN JSON_VALUE("t"."fault_properties", '\$fault_type') = 'fault_3' THEN 'fast charging' WHEN JSON_VALUE("t"."fault_properties", '\$fault_type') = 'fault_4' THEN 'low battery' WHEN JSON_VALUE("t"."fault_properties", '\$fault_type') = 'fault_5' THEN 'charging issue' WHEN JSON_VALUE("t"."fault_properties", '\$fault_type') = 'fault_6' THEN 'engine overheating' END</pre>
Documentation	
Type	String
Add URL	
<input type="button" value="Cancel"/> <input type="button" value="Save"/>	

e. Now, let's rename the **subtype** dimension into **Firmware version** :

Edit dimension

Name	<input type="text" value="Firmware version"/>
Description	<input type="text"/>
	<input checked="" type="radio"/> Basic <input type="radio"/> Custom
Column	<input type="text" value="__subtype"/>
Type	<input type="text" value="String"/>
Add URL	
<input type="button" value="Cancel"/> <input type="button" value="Save"/>	

f. Now let's add/modify some custom measures to the datacube.

← IoT Ev Final ⚙

Edit data cube

General	Measures	Cancel	Save
<input checked="" type="checkbox"/> Dimensions	<input type="text" value="Number of Events"/> COUNT(*)	<input type="button"/>	<input type="button"/>
<input type="checkbox"/> Measures	<input type="text" value="Event Timing"/> SUM(L."event_timing")	<input type="button"/>	<input type="button"/>
<input type="checkbox"/> Access	<input type="text" value="Defect Ratio"/> 100.0 * COUNT(*) FILTER (WHERE CASE WHEN JSON_VALUE("\$.fault_properties", '\$.fault_type') in ('fault_B', 'fault_B') THEN '1' else '0' END) / COUNT(*)	<input type="button"/>	<input type="button"/>
<input type="checkbox"/> Access filters	<input type="text" value="Torque"/> Torque Achieved (Avg) AVG(L."torque_achieved") Torque Achieved (Max) MAX(L."torque_achieved") Torque Available Regen (Avg) AVG(L."torque_available_regen") Torque Available Regen (Max) MAX(L."torque_available_regen") Torque Available Motoring (Avg) AVG(L."torque_available_motoring") Torque Available Motoring (Max) MAX(L."torque_available_motoring") Temperature Motor Temperature (Avg) AVG(L."motor_temperature") Motor Temperature (Max) MAX(L."motor_temperature"-100)	<input type="button"/>	<input type="button"/>
<input type="checkbox"/> Advanced		<input type="button"/>	<input type="button"/>

- for each **Torque** and **Temperature** measure, only keep/add an **AVG** and a **MAX** expression :

		Motor Temperature (Avg) AVG(t."motor_temperature")
↳	Torque Achieved (Avg) AVG(t."torque_achieved")	Motor Temperature (Max) MAX(t."motor_temperature"-100)
↳	Torque Achieved (Max) MAX(t."torque_achieved")	Temperature IGBT A (Avg) AVG(t."temperature_IGBT_A")
↳	Torque Available Regen (Avg) AVG(t."torque_available_regen")	Temperature IGBT A (Max) MAX(t."temperature_IGBT_A"-100)
↳	Torque Available Regen (Max) MAX(t."torque_available_regen")	Temperature IGBT B (Avg) AVG(t."temperature_IGBT_B")
↳	Torque Available Motoring (Avg) AVG(t."torque_available_motoring")	Temperature IGBT B (Max) MAX(t."temperature_IGBT_B"-100)
↳	Torque Available Motoring (Max) MAX(t."torque_available_motoring")	Temperature IGBT C (Avg) AVG(t."temperature_IGBT_C")
		Temperature IGBT C (Max) MAX(t."temperature_IGBT_C"-100)

- g. finally , add a last measure named “**Total Energy added**” with following SQL expression:

```
SUM(JSON_VALUE("t"."charging_properties",'$.total_charge_energy_added'))
```

Edit measure

Time Period	Value
Last minute	20.94 k
Last hour	1.41 m
Last day	33.45 m

General

Format

Advanced

Basic **Custom**

Formula

```
SUM(JSON_VALUE("t"."charging_properties",'$.total_charge_energy_added'))
```

[Documentation](#)

Cancel **Save**

- h. Give your datacube a name. And click “Save” :

General

Dimensions

Measures

Access

Access filters

Advanced

Name: iot Ev

Description

Source: From table

Default timezone: Etc/UTC

Minimum auto-refresh rate: None

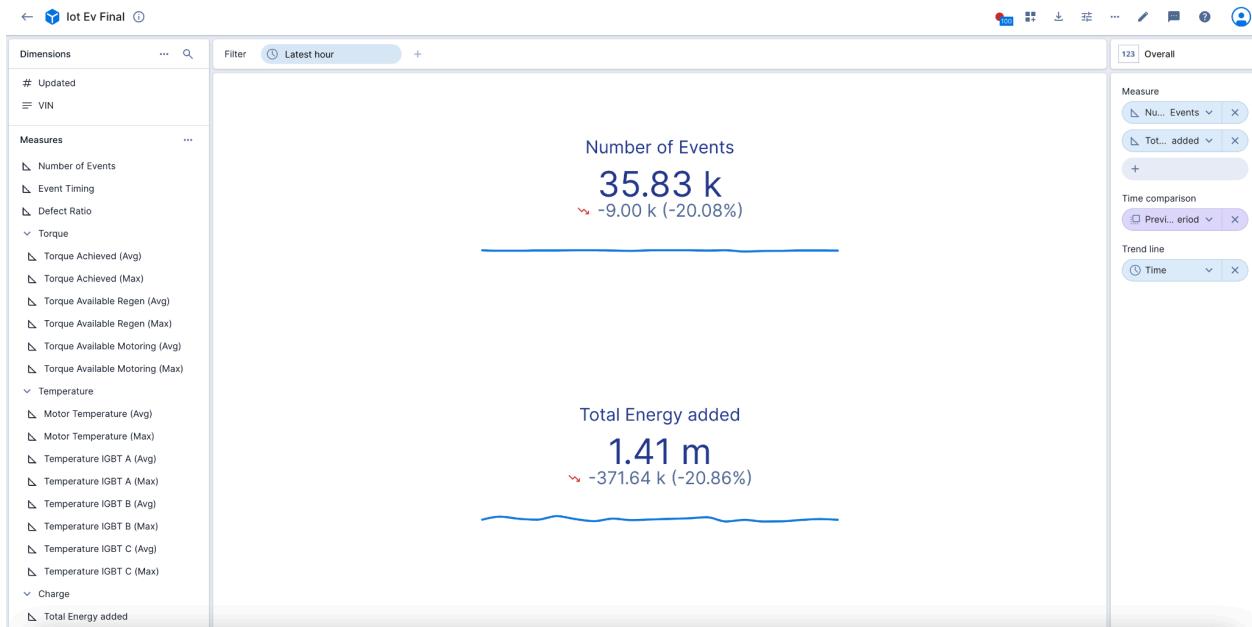
Primary time dimension: Time

Query timeout override

Customize data cube UI colors

Cancel **Save**

- i. Now you can use the UI to drag and drop and analyze the data with existing dimensions and measures, for example, drag “Total Energy added” onto the measure lists. You can do this with multiple measures and change the measures according to your need. From here you can click  at the top right, to add this to a dashboard, but we’ll get into that further in a second. Go ahead and try it. We’ll get into creating dashboards next.

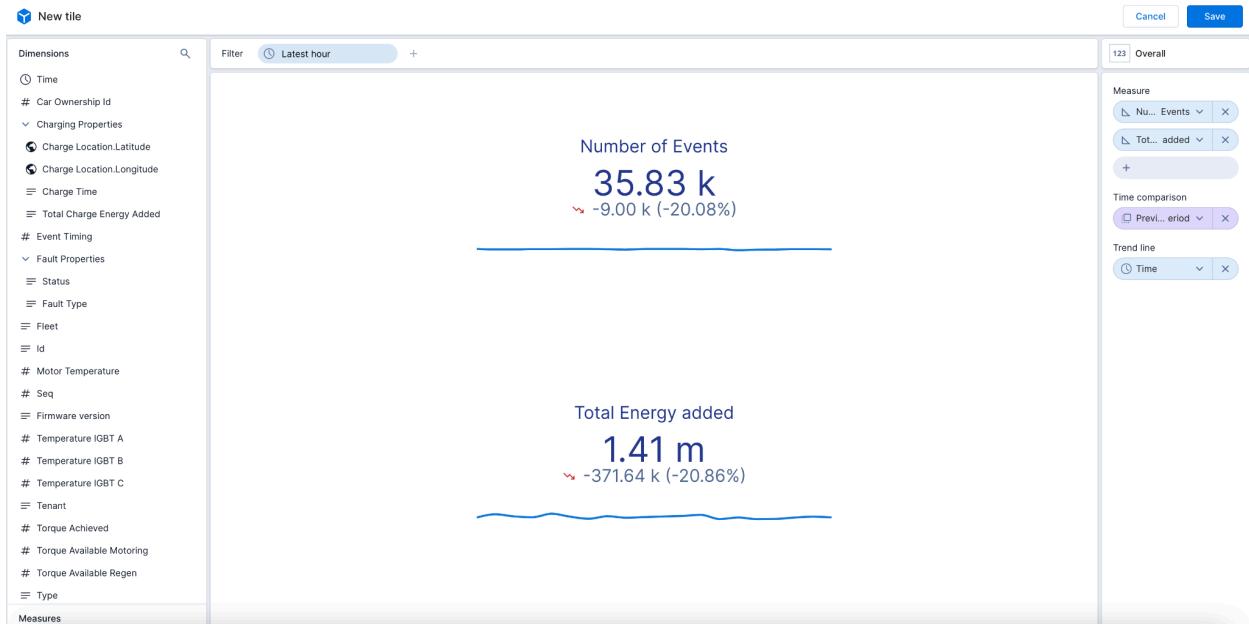


2. Creating a dashboard from a datacube is easy:

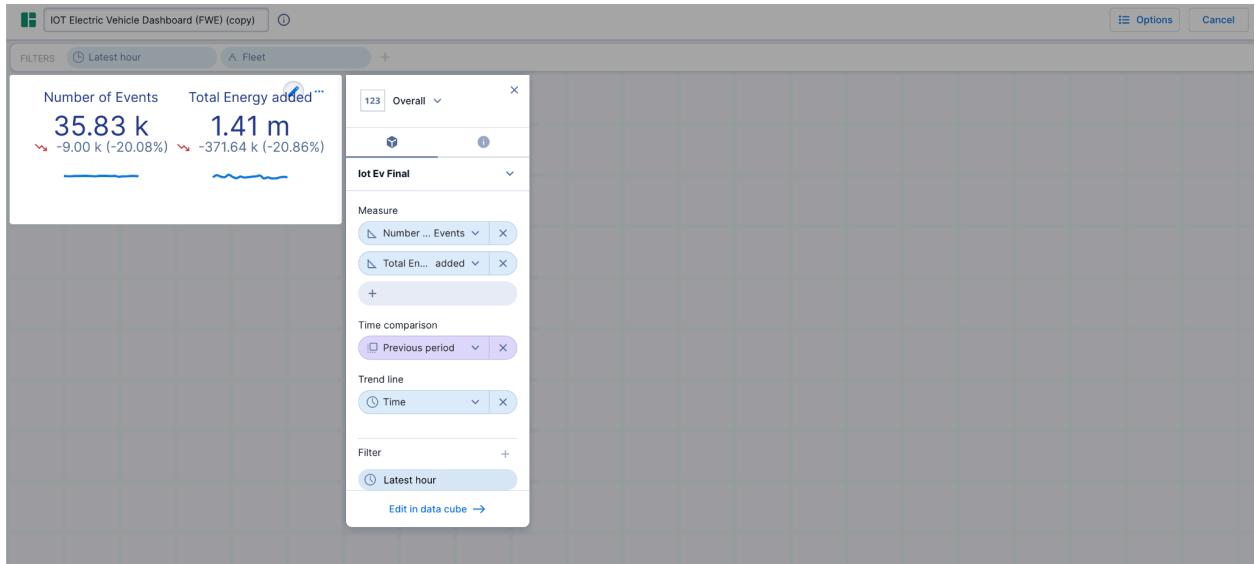
- Navigate back to the home screen and click “Dashboards” and “Create dashboard” at the top. Name the dashboard as “IOT Dashboard”



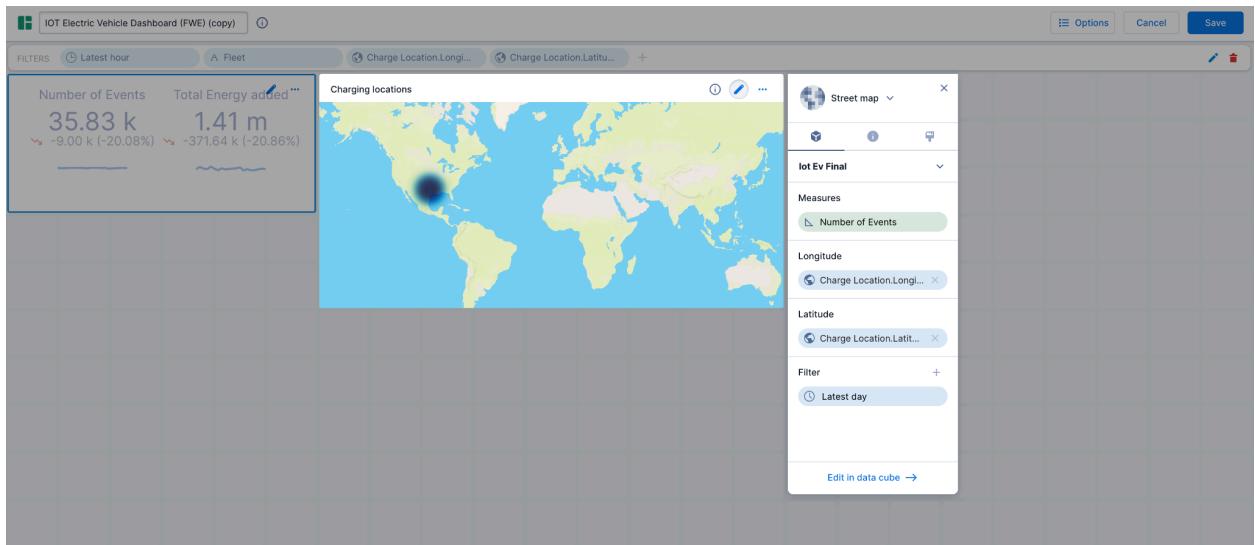
- Click and select an area on the dashboard, in the drop down select the cube you just created, it may be selected by default if you only have one cube in your polaris project. For chart type, select “Overall”, and click “Edit in data cube”, just create the same Overall visualization as previous step, then save :



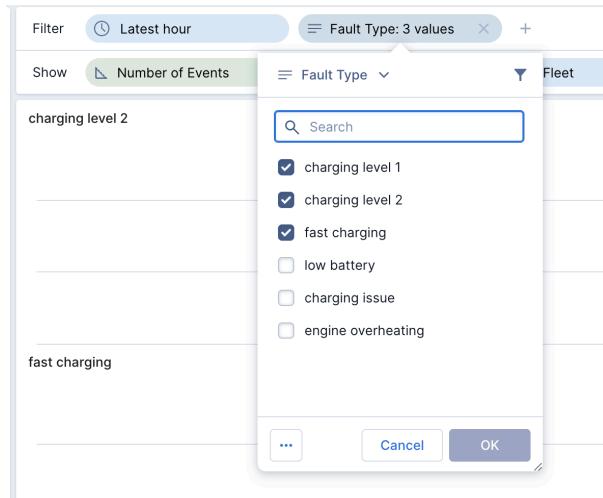
Visualization is added to the dashboard :



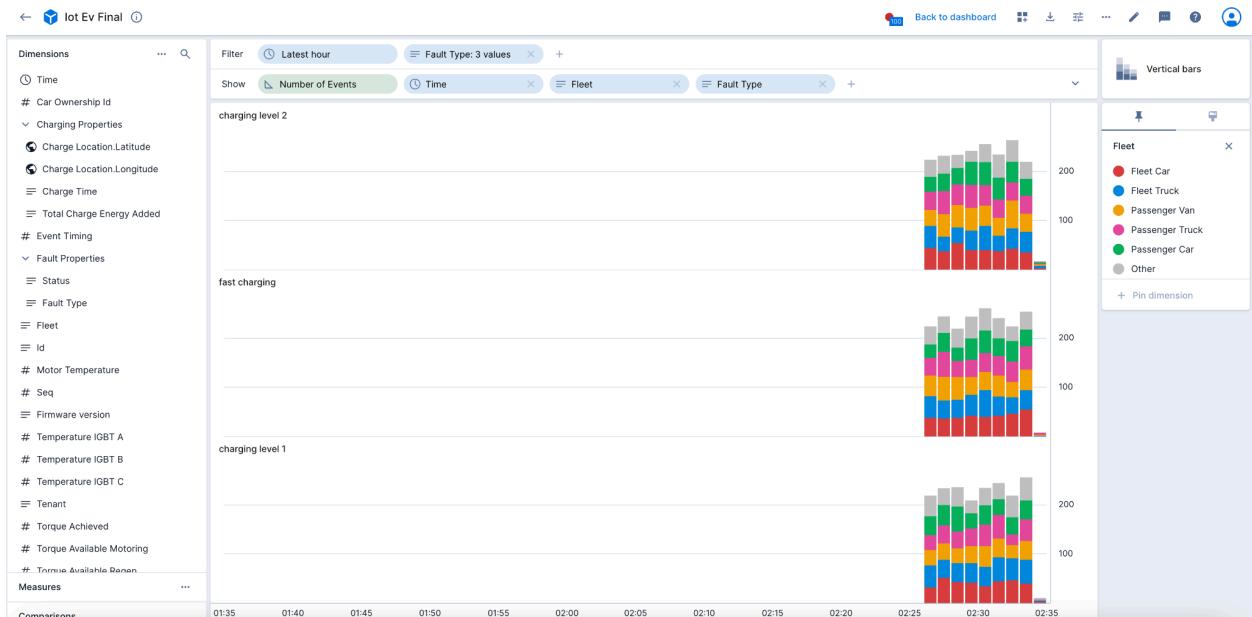
- c. Now, let's create a new chart against the same datacube, select the **street map** visualization, it will automatically propose to use the 2 existing geo dimensions **LAT** and **LON** :



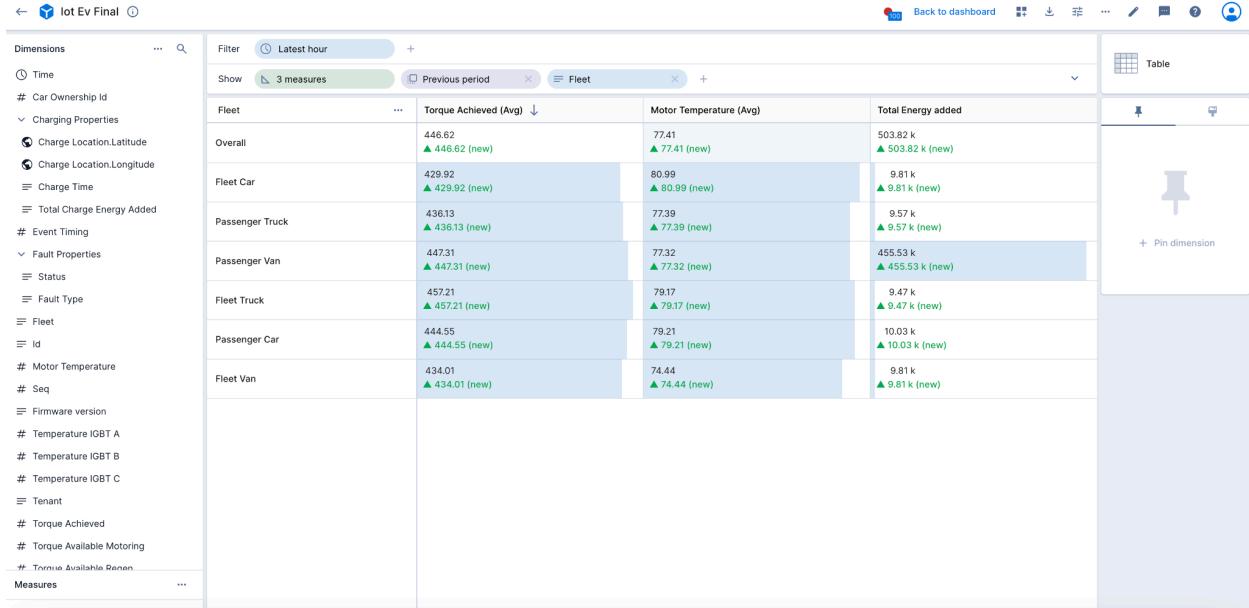
- d. Now, let's create a new chart on our dashboard against the same datacube to monitor the charging status per fleet type, select vertical bars chart type, then add **time**, **fleet** and **fault type**, also add a filter on **fault type** restricted to the 3 charging types below :



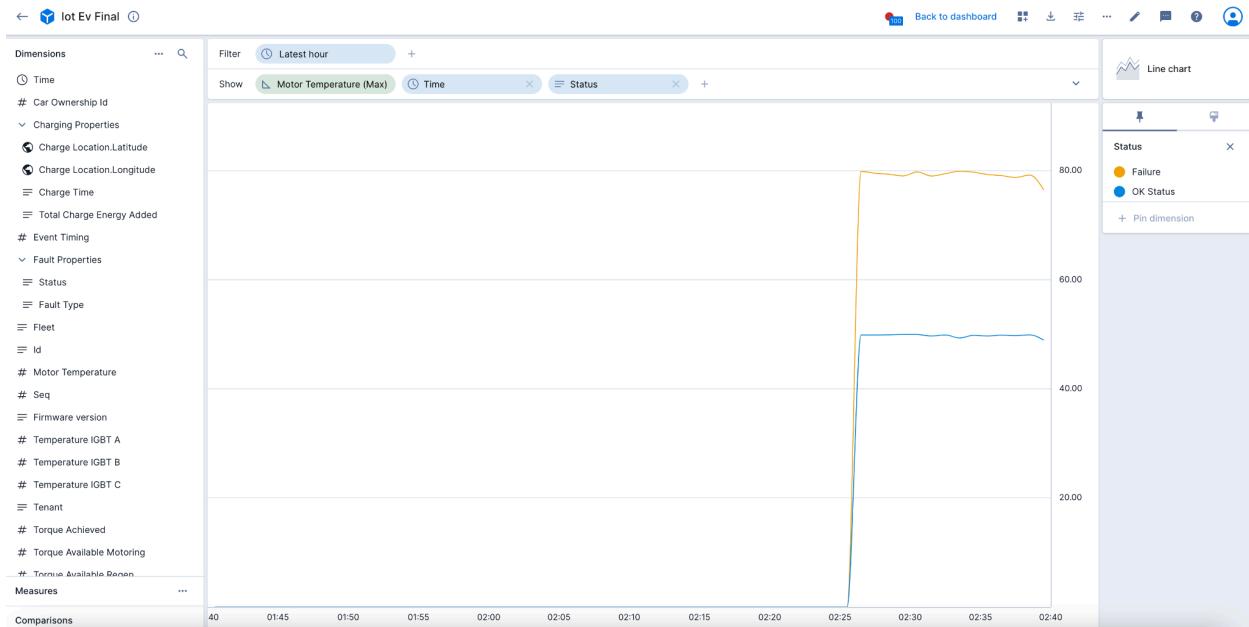
Your report should look like this before you add it to the dashboard :



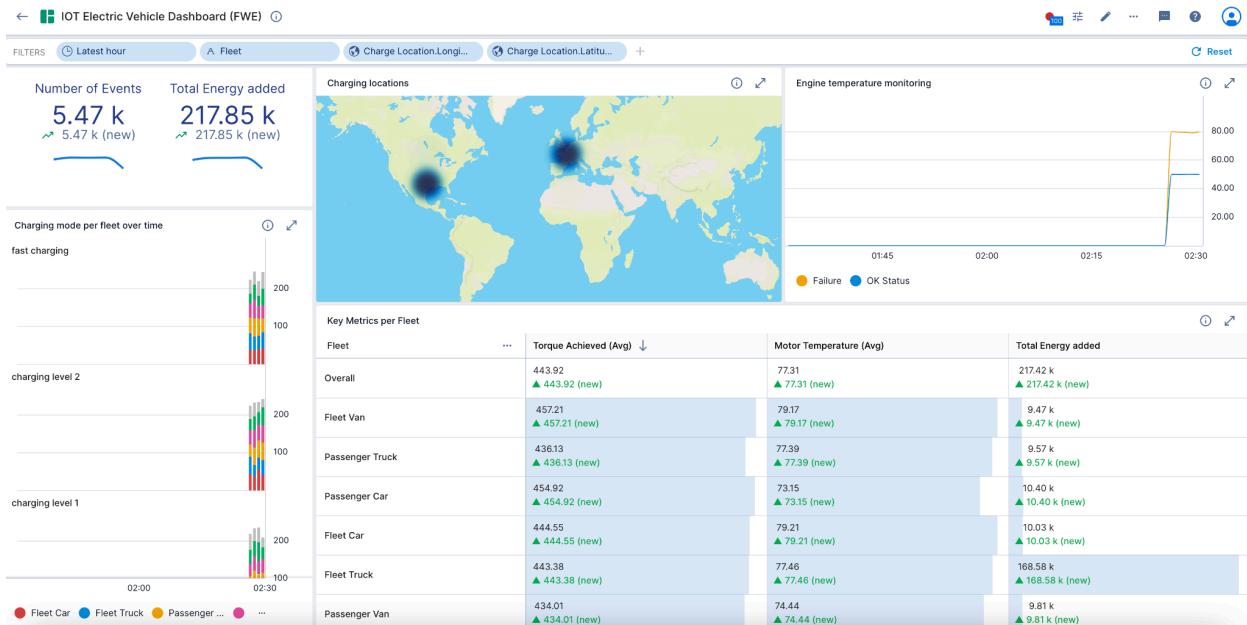
- e. Now let's add a new chart to monitor Avg Torque, Avg Motor Temperature and Total energy added by selecting a Table visualization, adding these 3 measures and adding the previous period for comparison, report should look like this before you add it to the dashboard :



- f. Finally, let's add one final chart to the dashboard to measure the Max Motor temperature per Status by selecting a Line chart visualization, and adding **Motor Temperature (Max)**, **Time** and **Status** to the show bar, your report should look like this before you add it to the dashboard :



- g. Finally rearrange your tiles to fit the screen, edit predefined filters (time to filter on latest hour only, fleet) and save the dashboard. It should look like the example below. You can now click and interact with the dashboard, and filter it. You can also share the dashboard URL with other users, as you filter it the URL preserves the state of the filter. You can click the “Reset” button at the top right and revert to the unfiltered version. For more tips and trick see <https://docs.imply.io/polaris/create-a-dashboard>



3. For instructions on how to create alerts see: <https://docs.imply.io/polaris/set-up-alerts/>
4. For instructions on how to create embedded visualizations see: <https://docs.imply.io/polaris/embed-visualizations>

General documentation around querying polaris via the API and all the building blocks is all available here.

<https://docs.imply.io/polaris/query>

<https://docs.imply.io/polaris/api-query>

<https://docs.imply.io/api/polaris/api-reference>