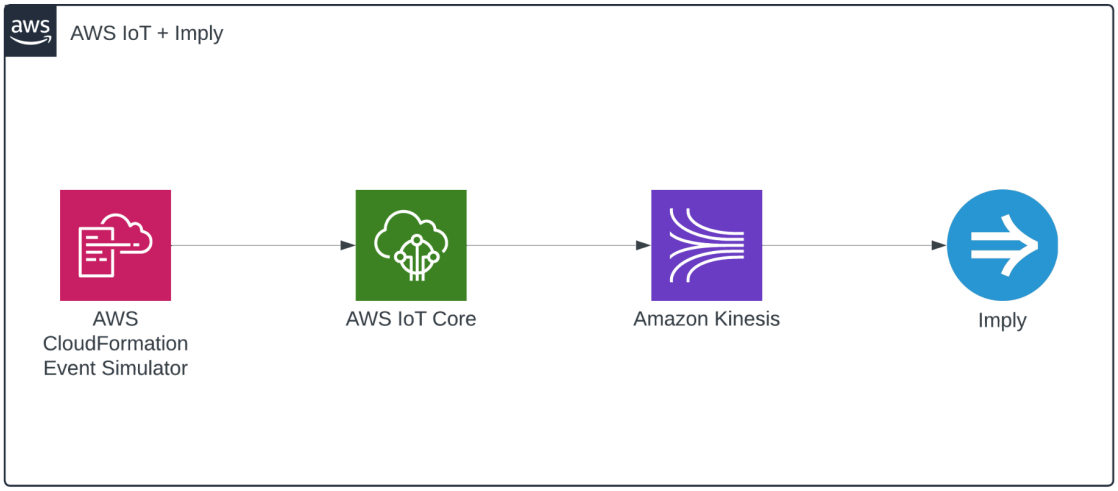
IoT Events Workshop | End-to-End Setup

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.

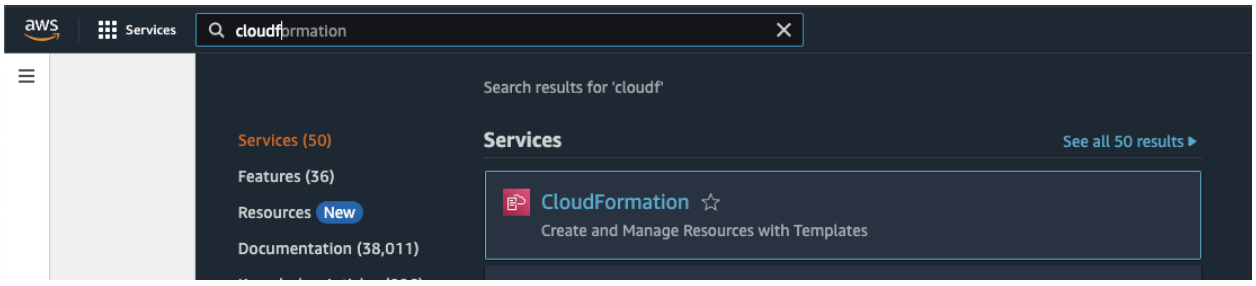


Step 0 - Prerequisites

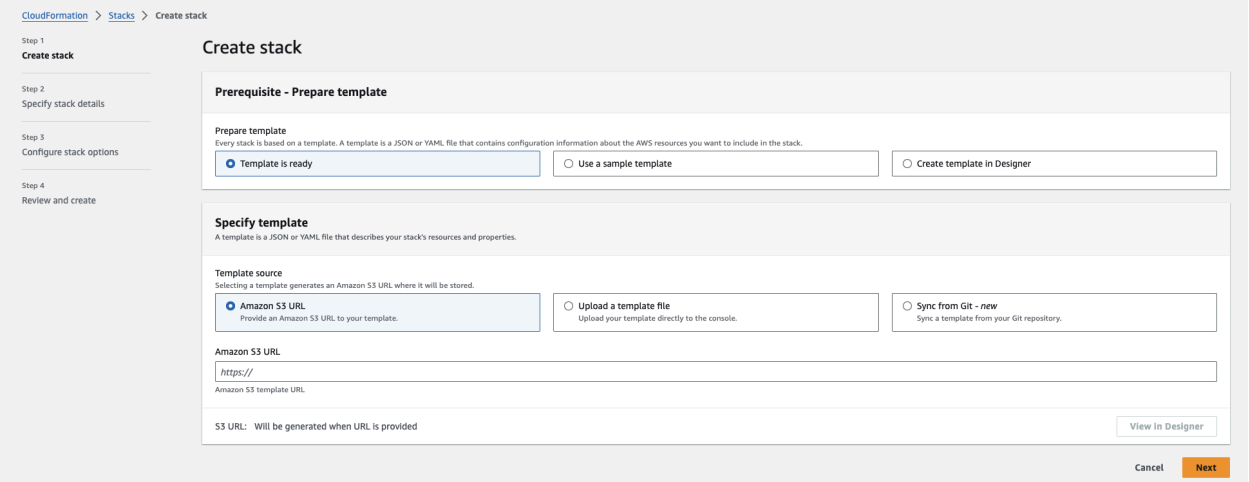
1. Make sure you have permissions to manage and create resources on your AWS project
2. Follow the steps to setup an Imply Polaris free trial account and sign-in for the first time
    a. Select a region for the Imply project that ideally match (this is not mandatory)
    b. us-east-1 is a good one to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

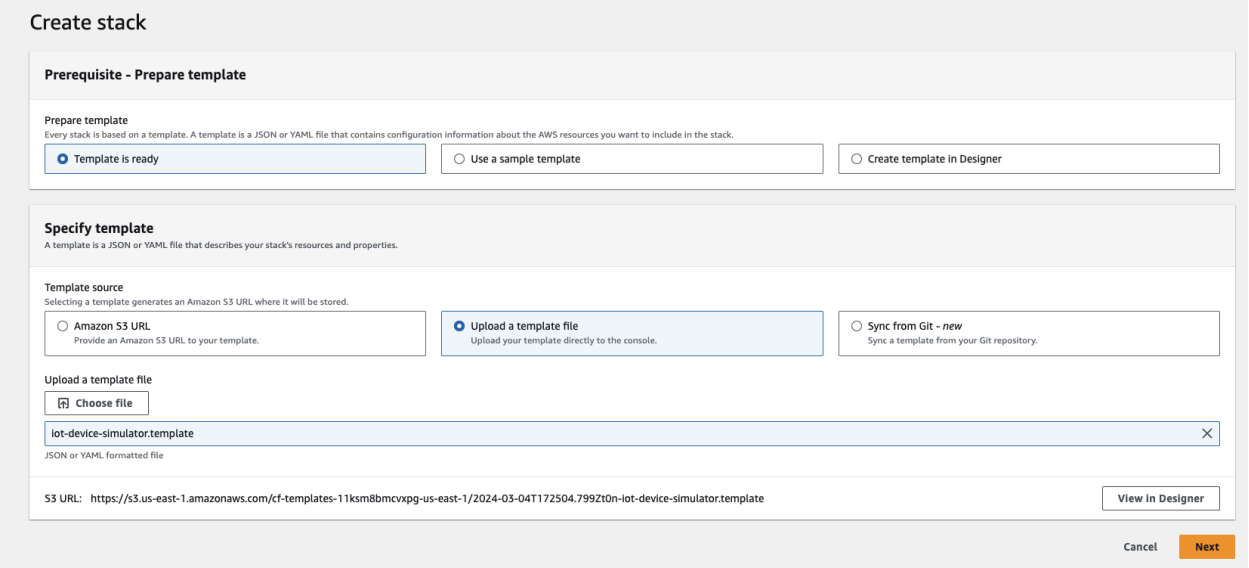


2. Click the “Create stack” button, your page should look like this

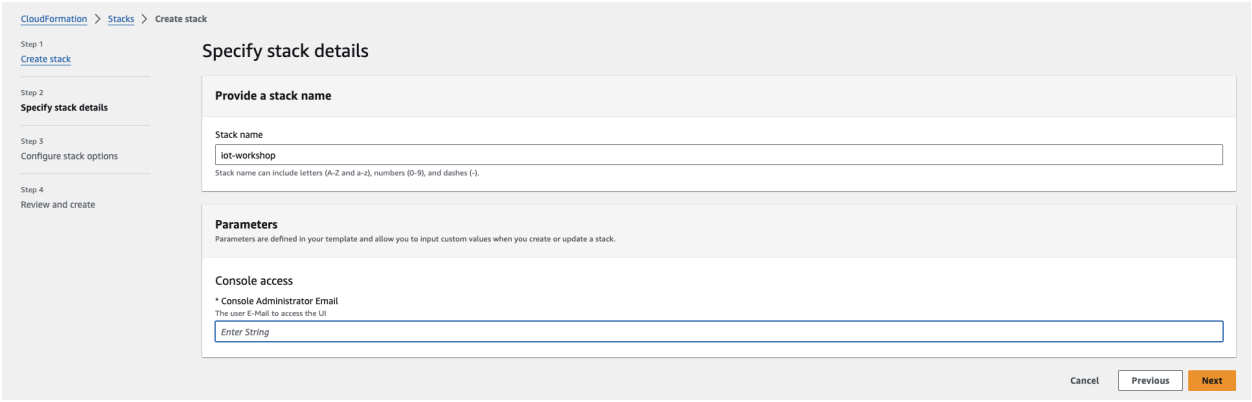


3. Download the latest data generator template from here by clicking the ‘View template” button, we will upload this template to the stack we are creating, then click “Next” https://docs.aws.amazon.com/solutions/latest/iot-device-simulator/template.html

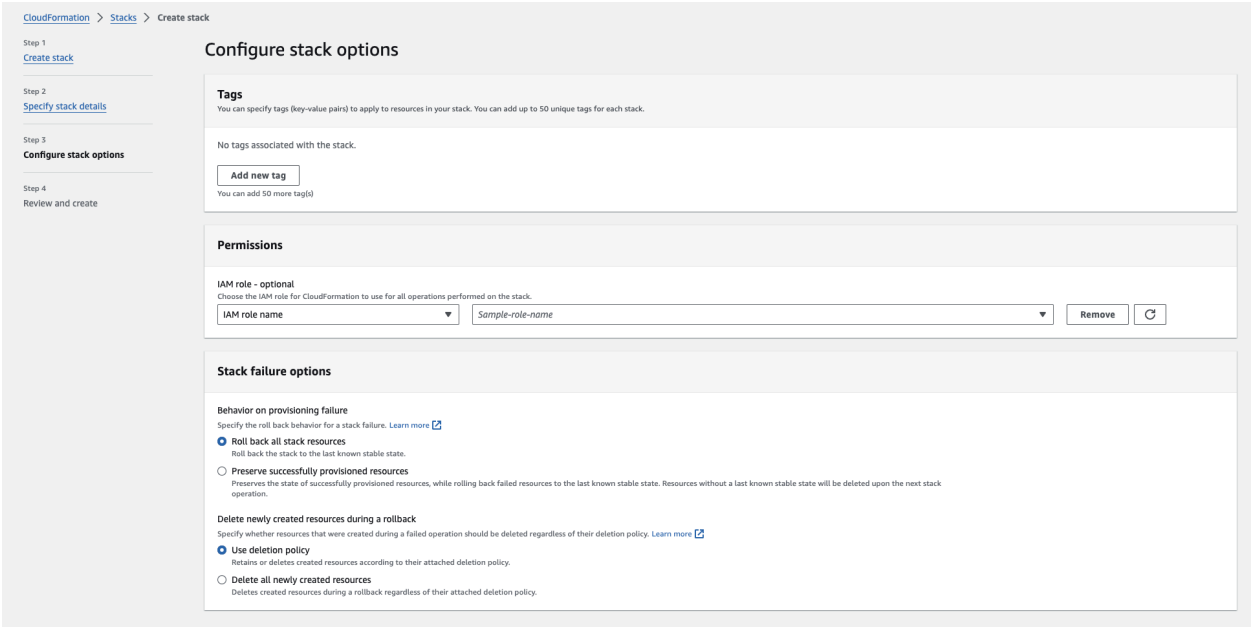
4. Click the “Upload a template file” option and then “choose file”, picking the template we just downloaded from the previous step, then click “Next” and



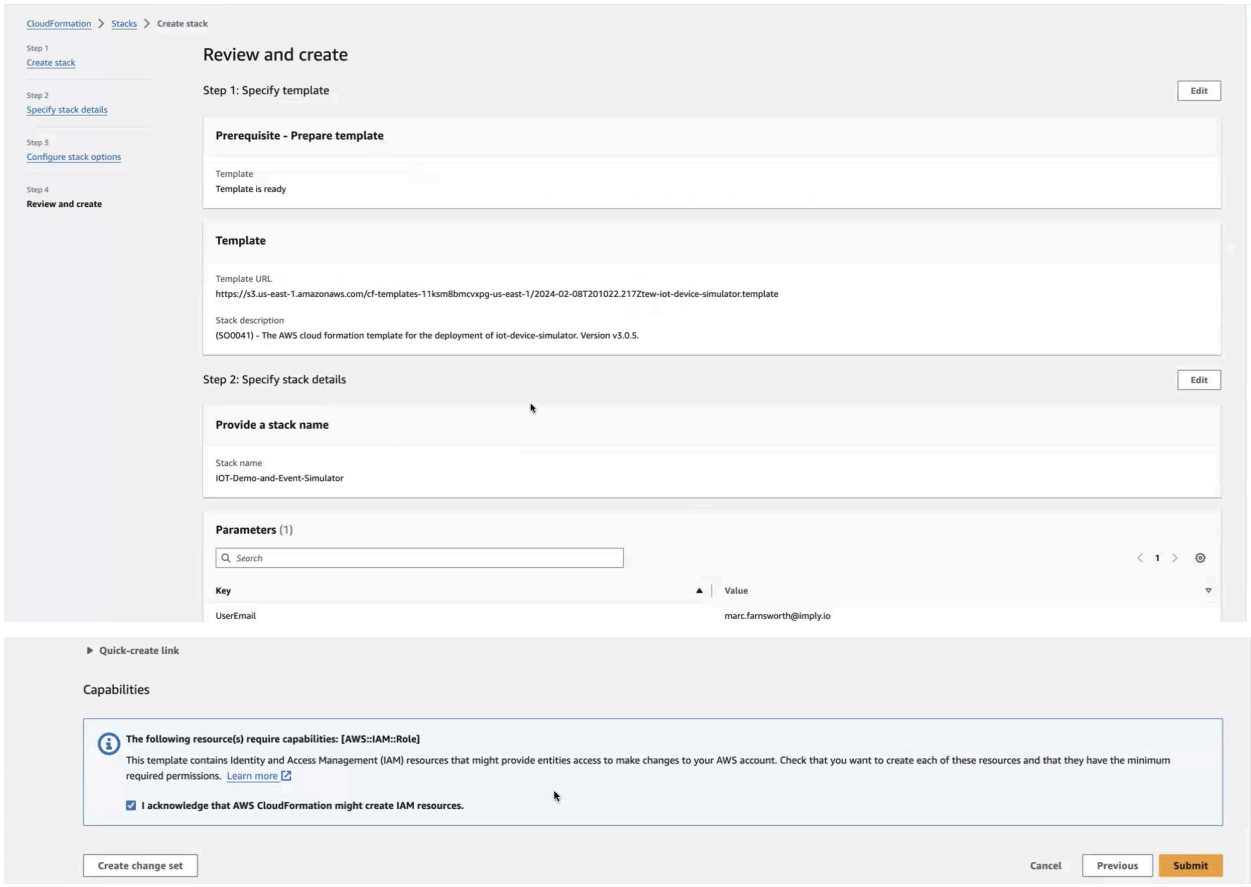
5. Specify your “Stack name”, then enter your email address for the “Console Administrator Email”, then click “Next”



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



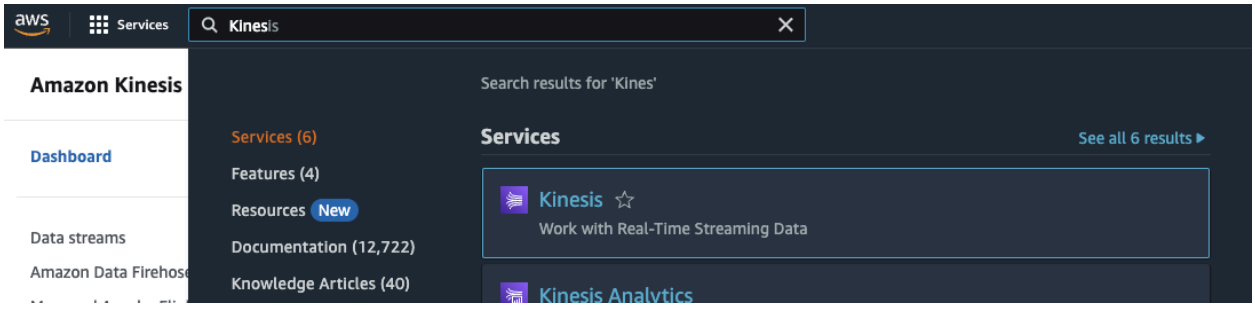
7. On the next page, “Review and create”, check the box “I acknowledge that AWS CloudFormation might create IAM resources.” at the bottom of the page, and click “Submit” to begin the stack creation



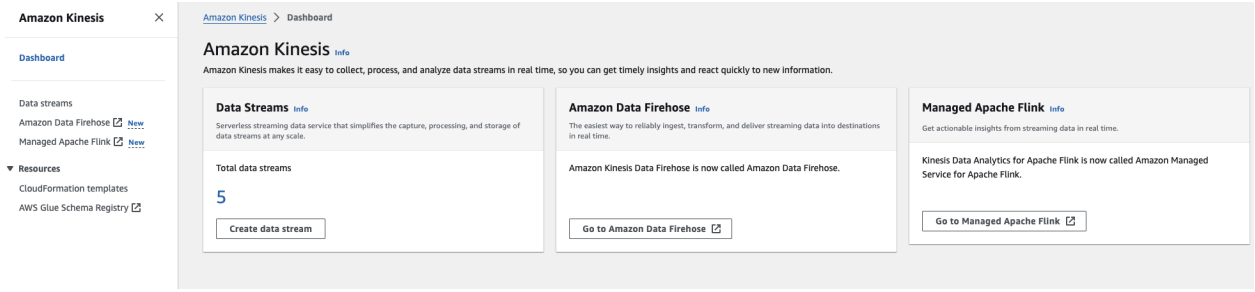
8. Creating the stack can take some some minutes to finish setup
9. Upon completion of the project, you will get an email from AWS inviting you to join your newly created simulator. The email will have a link should have a link to the environment, your username, and temporary password (which you will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

Step 2 - Setup of the Kinesis stream

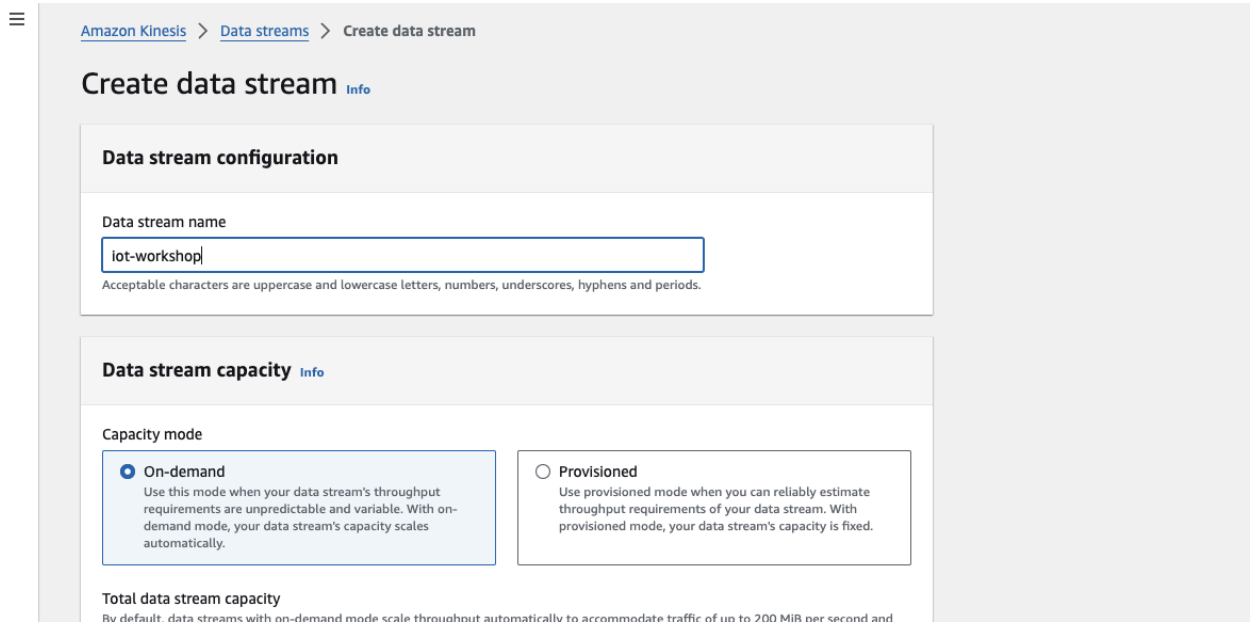
1. Search for “Kinesis” on the AWS search bar, and click “Kinesis”



2. Click “Create data stream”

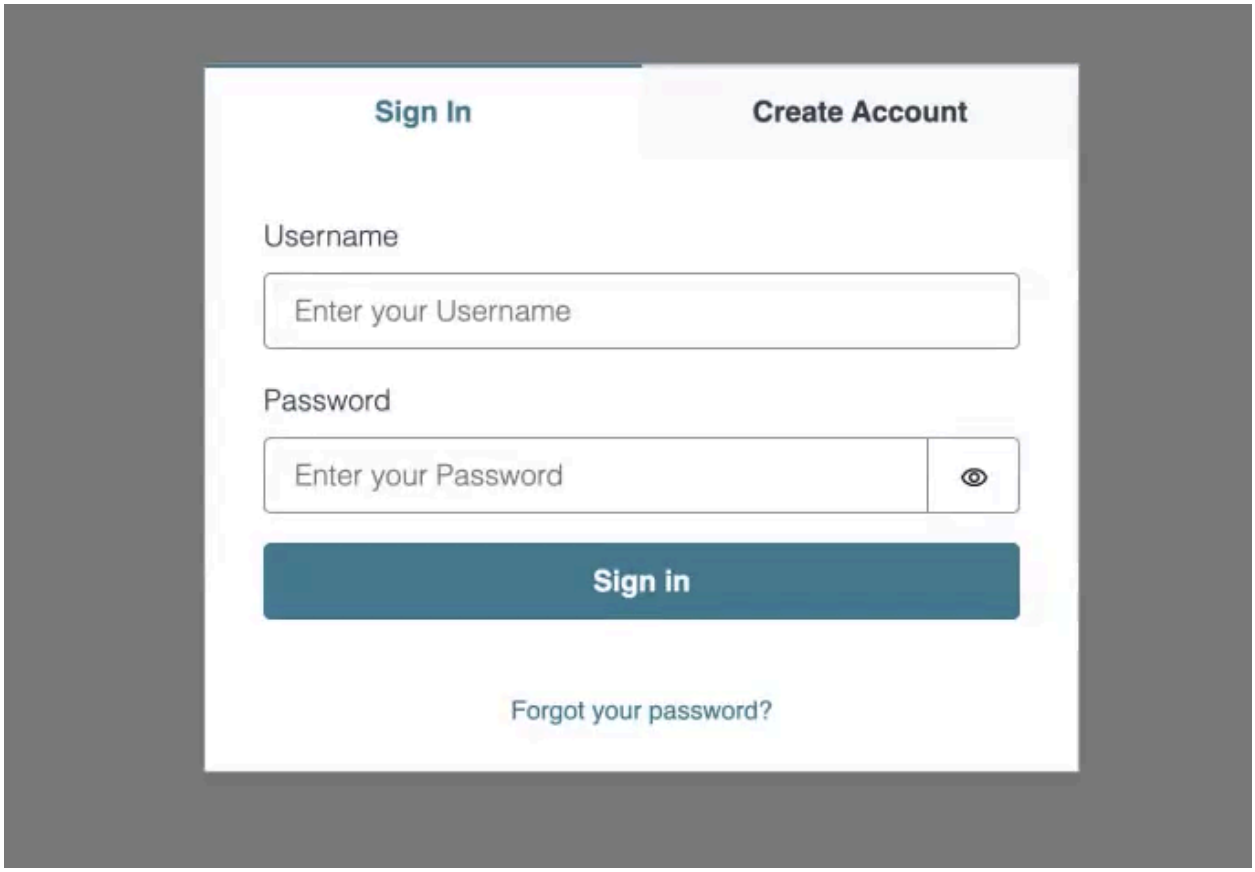


3. Provide a “Data stream name”, leave all other defaults, and click “Create data stream”

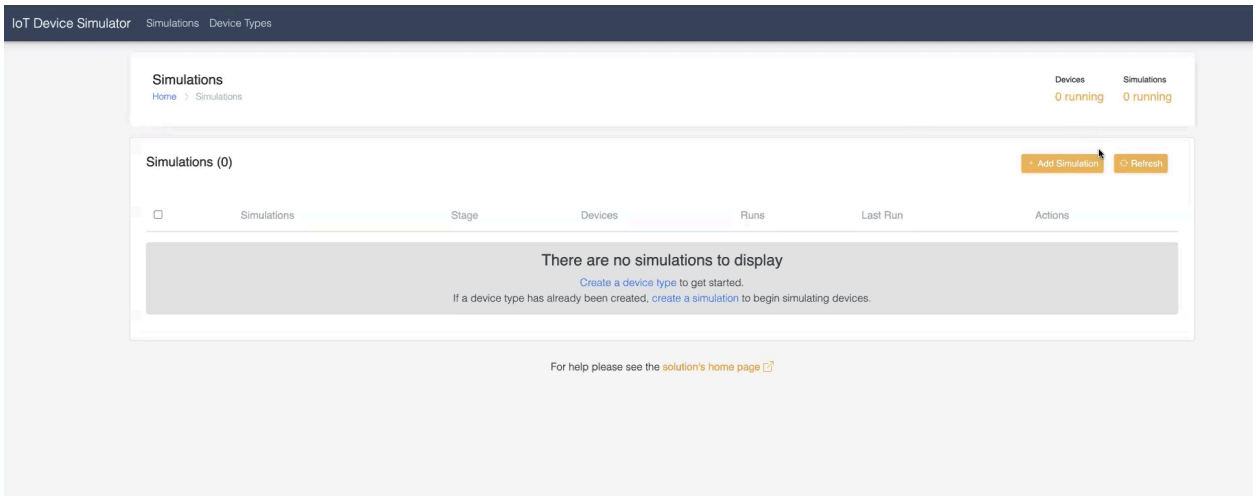


Step 3 - Create your data simulator

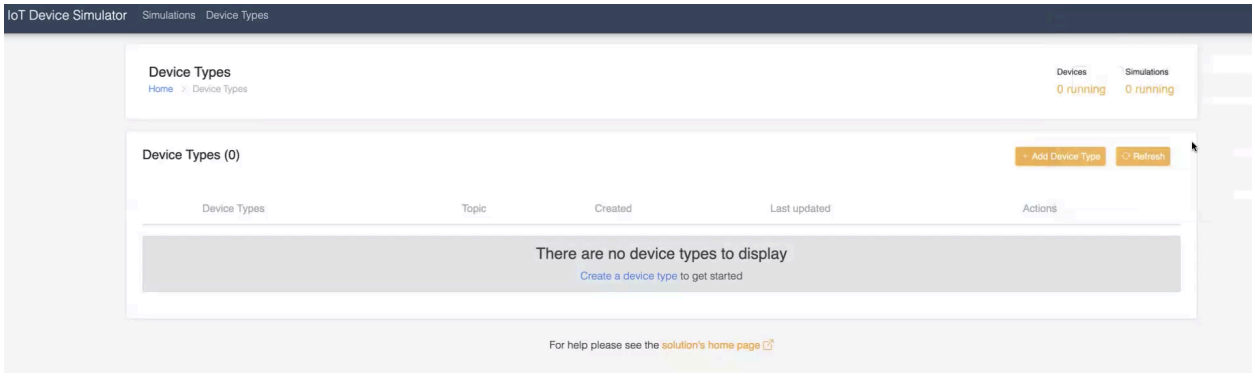
1. Login to your IoT Device Simulator with the username and temporary password that was emailed to you, and it will prompt you to change your password



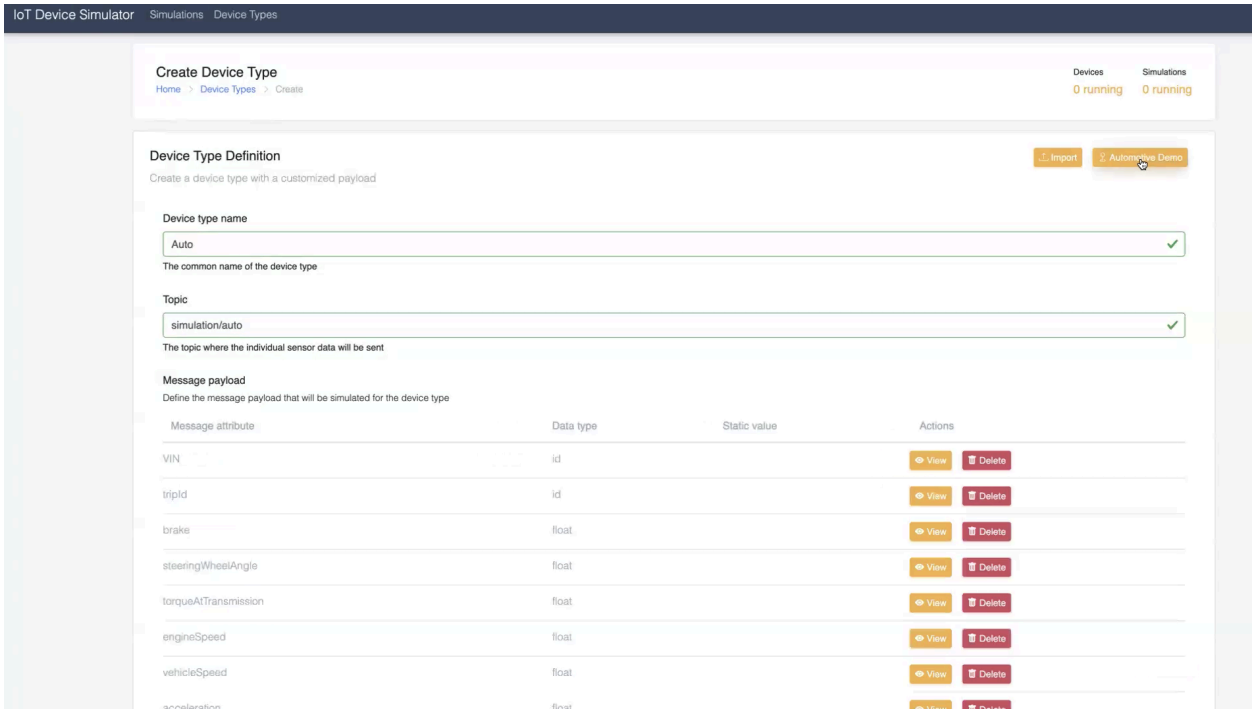
2. This should be what your landing page looks like, next click on “Device Types” at the top of the page

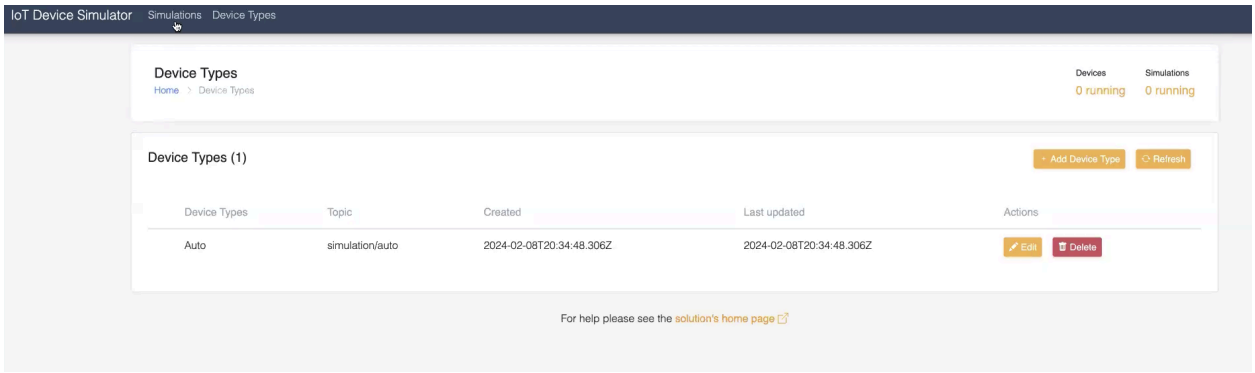


3. Click “ + Add Device Type”

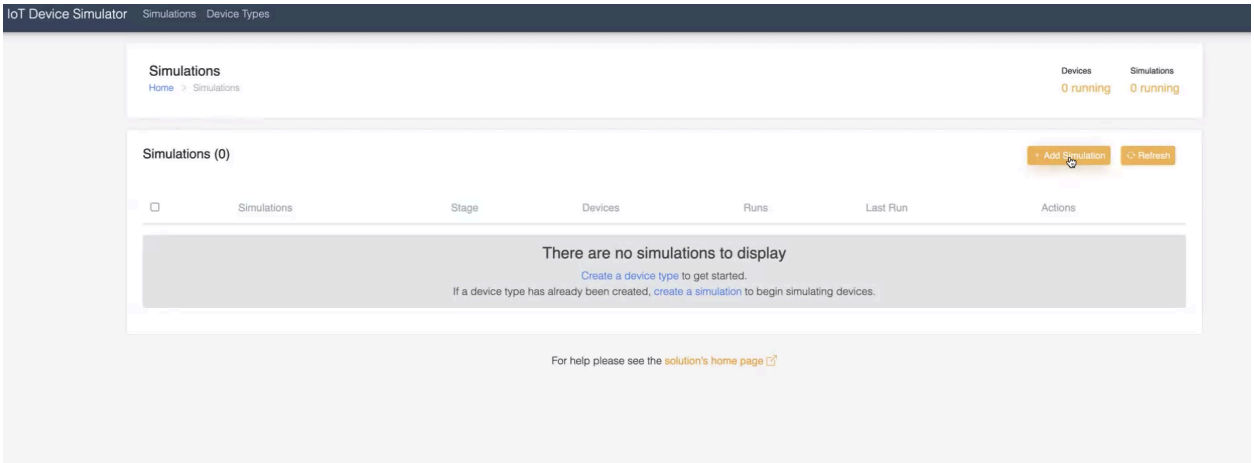


4. Fill out the “Device type name” as “Auto”, fill out the “Topic” as “simulation/auto” click “Automotive Demo” then click “Save”

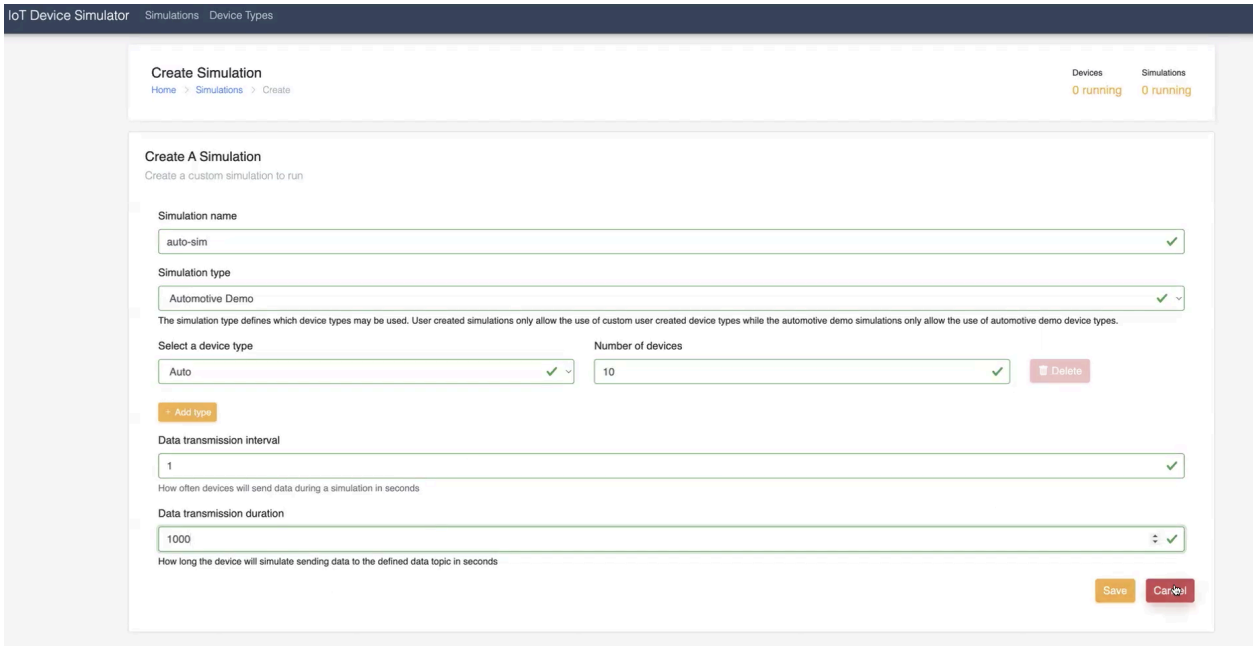


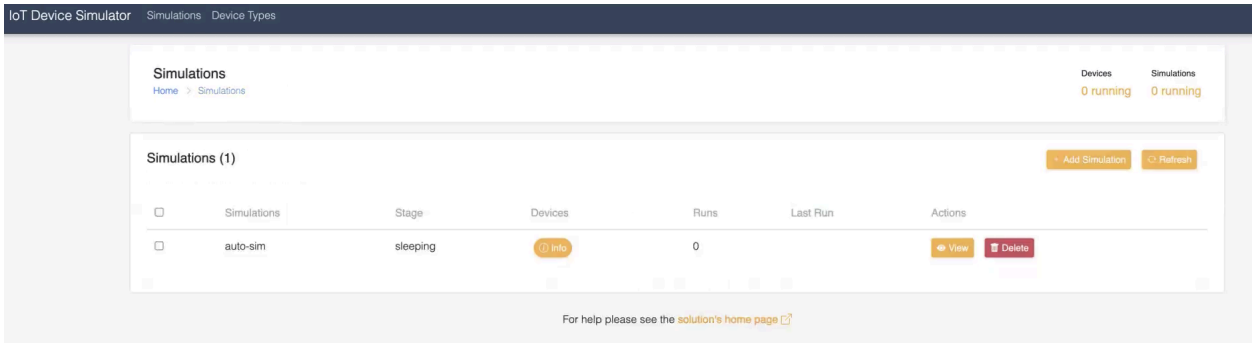


5. Click “Simulations” at the top left corner, click “+ Add Simulation”

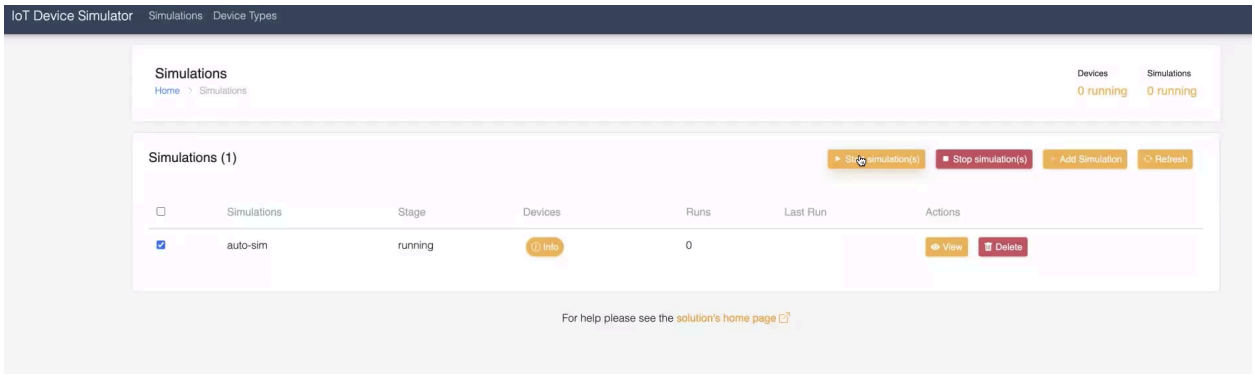


6. On the “Create a Simulation” page, name the simulation, select the “Automotive Demo” from the simulation type, select “Auto” from the device type, set the number of devices to 10, and change the “Data transmission duration” to 1000 and then save
    a. To use a custom evAuto, you can import this json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



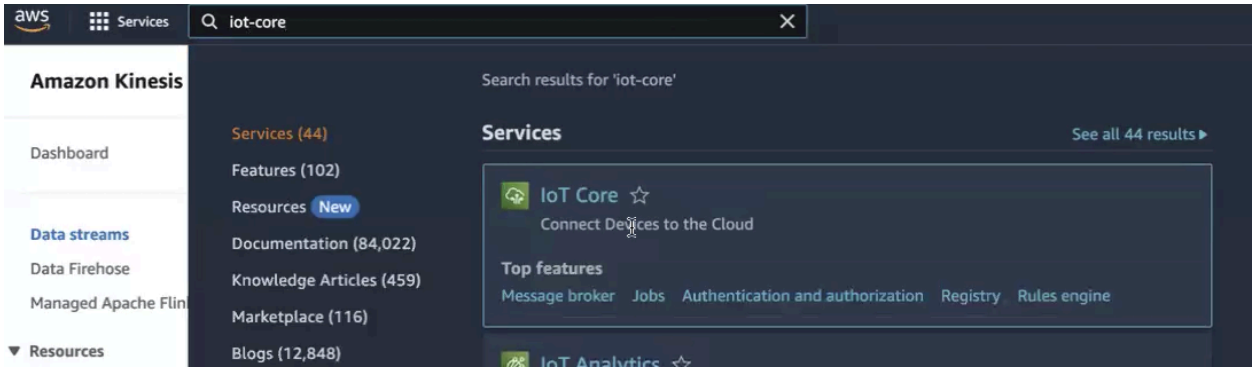


7. Check the box on the left of the newly created simulation, and click “Start simulation(s)”

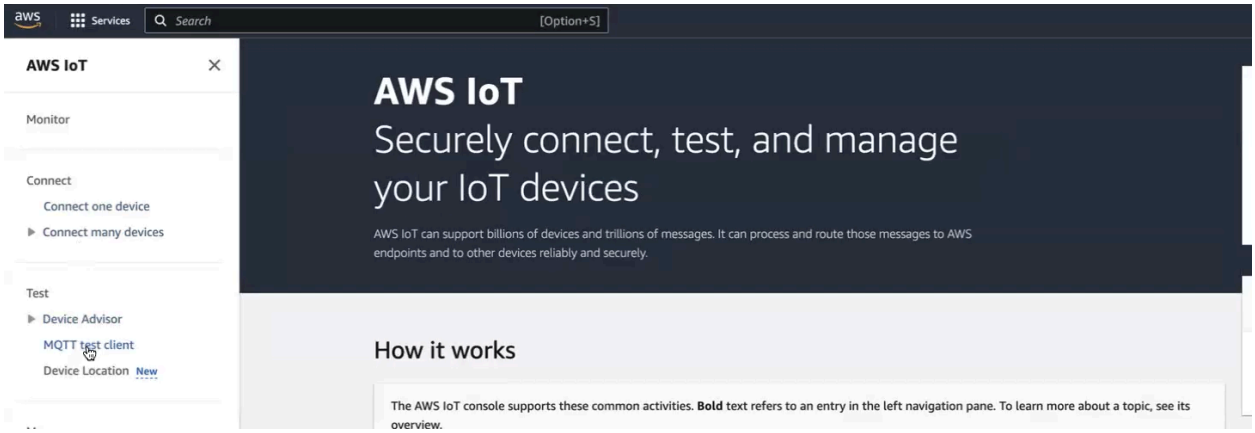


Step 4 - Connect the simulator to AWS IoT Core, and stream to Kinesis

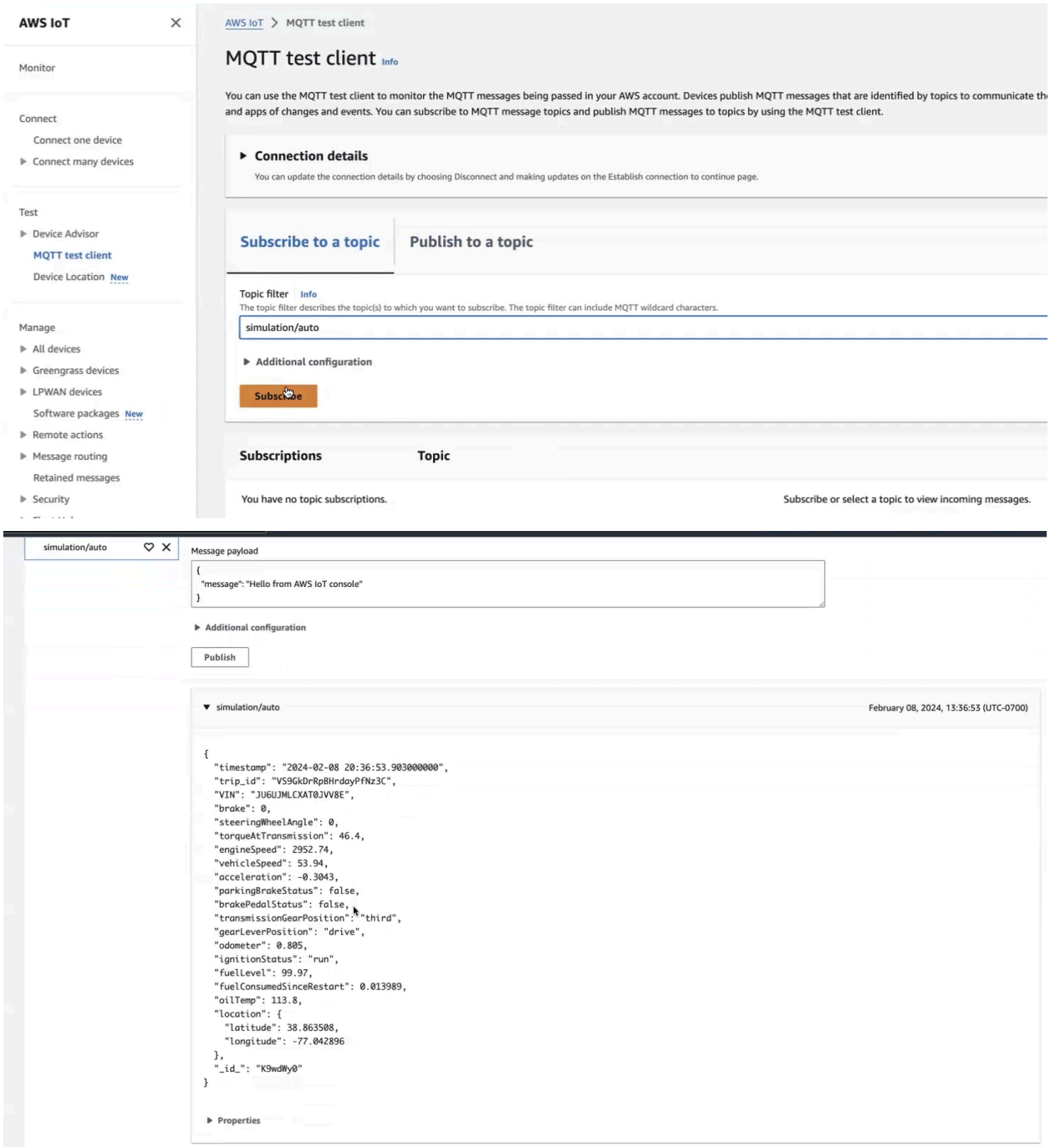
1. Search for “iot-core” and navigate there



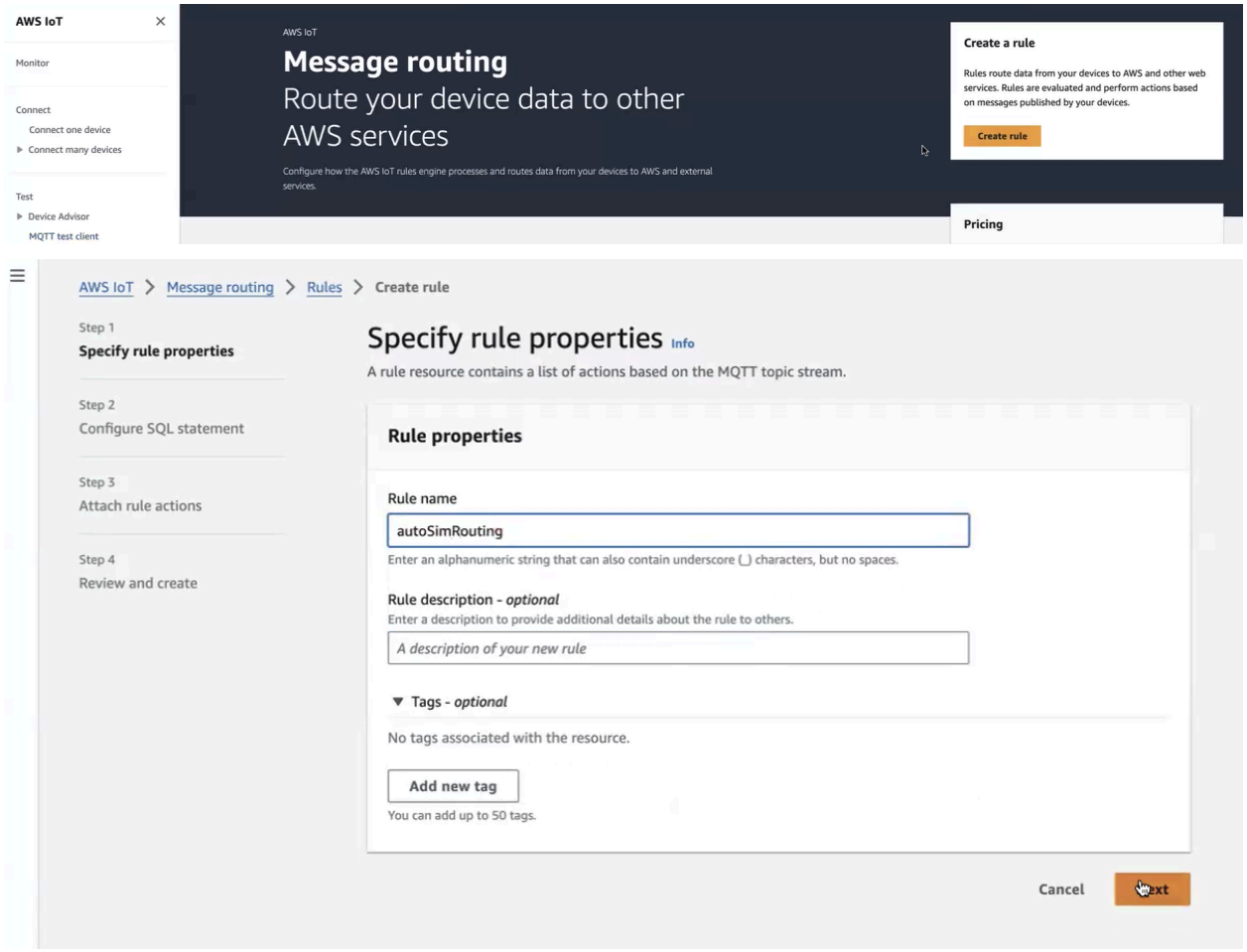
2. On the left menu click on “MQTT test client”



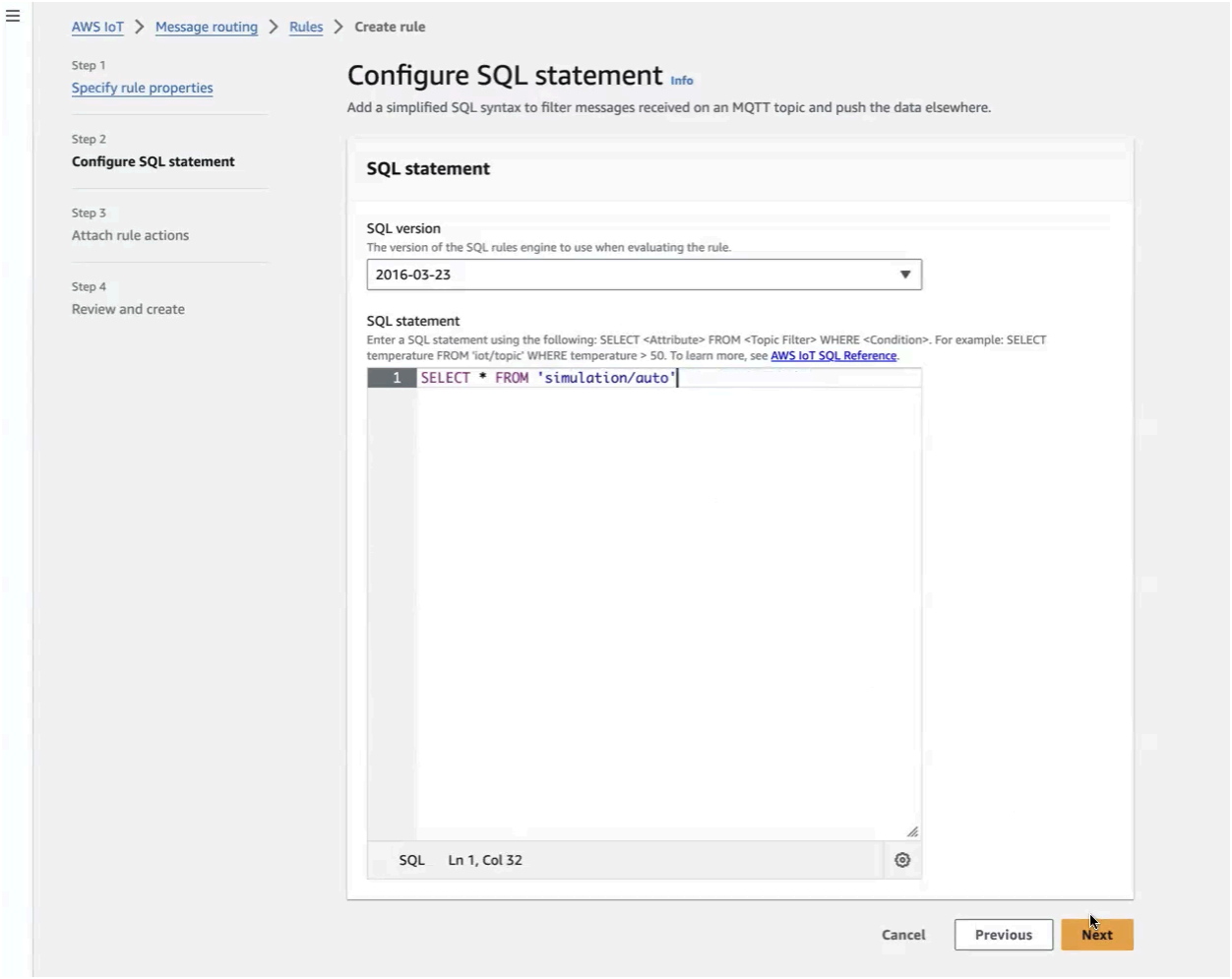
3. Test a subscription to the topic, enter “simulation/auto” as the topic filter, and click “Subscribe”, you should now see events coming through



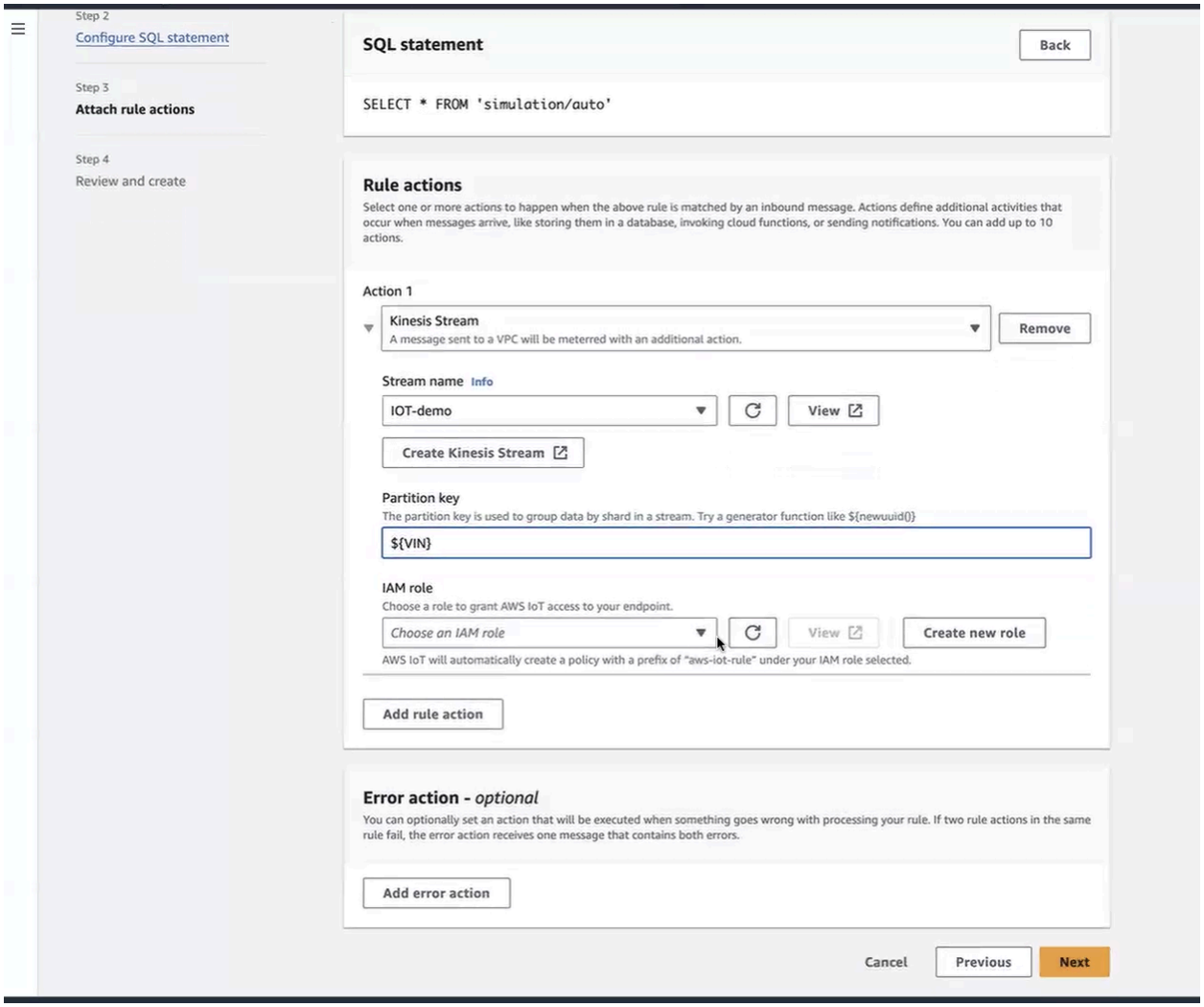
4. On the left menu, click on “Message routing”, click then “Create rule”, and provide a “Rule name” then click “Next”


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
Configure how the AWS IoT rules engine processes and routes data from your devices to AWS and external 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

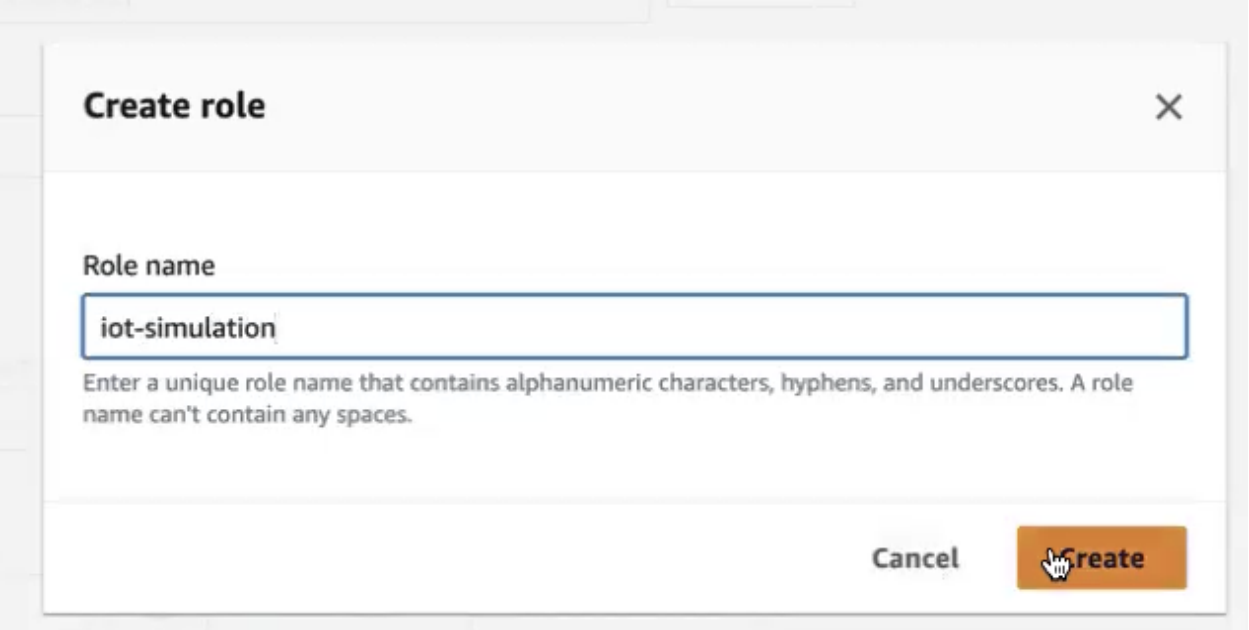
5. Configure the rule as “SELECT * FROM ‘simulation/auto’, then click “Next”



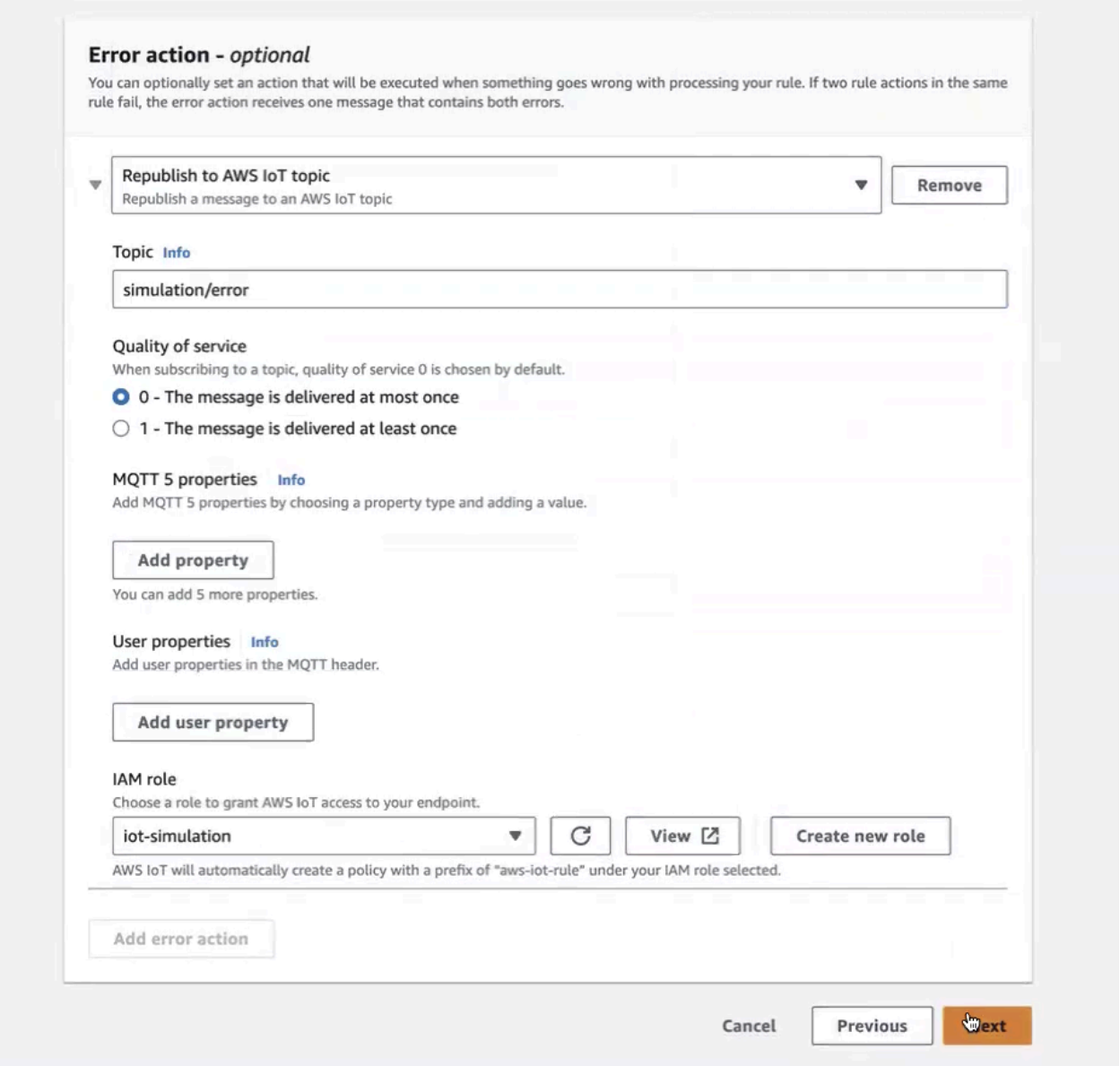
6. Configure for a “Kinesis stream” by selecting from the action dropdown, select the previously created stream “iot-workshop” from the “Stream name” dropdown, and add a “Partition key”,


Step 2
Configure SQL statement
Step 3
Attach rule actions
Step 4
Review and create
SQL statement
Back
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 meterred with an additional action.
Remove
Stream name Info
IOT-demo
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
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

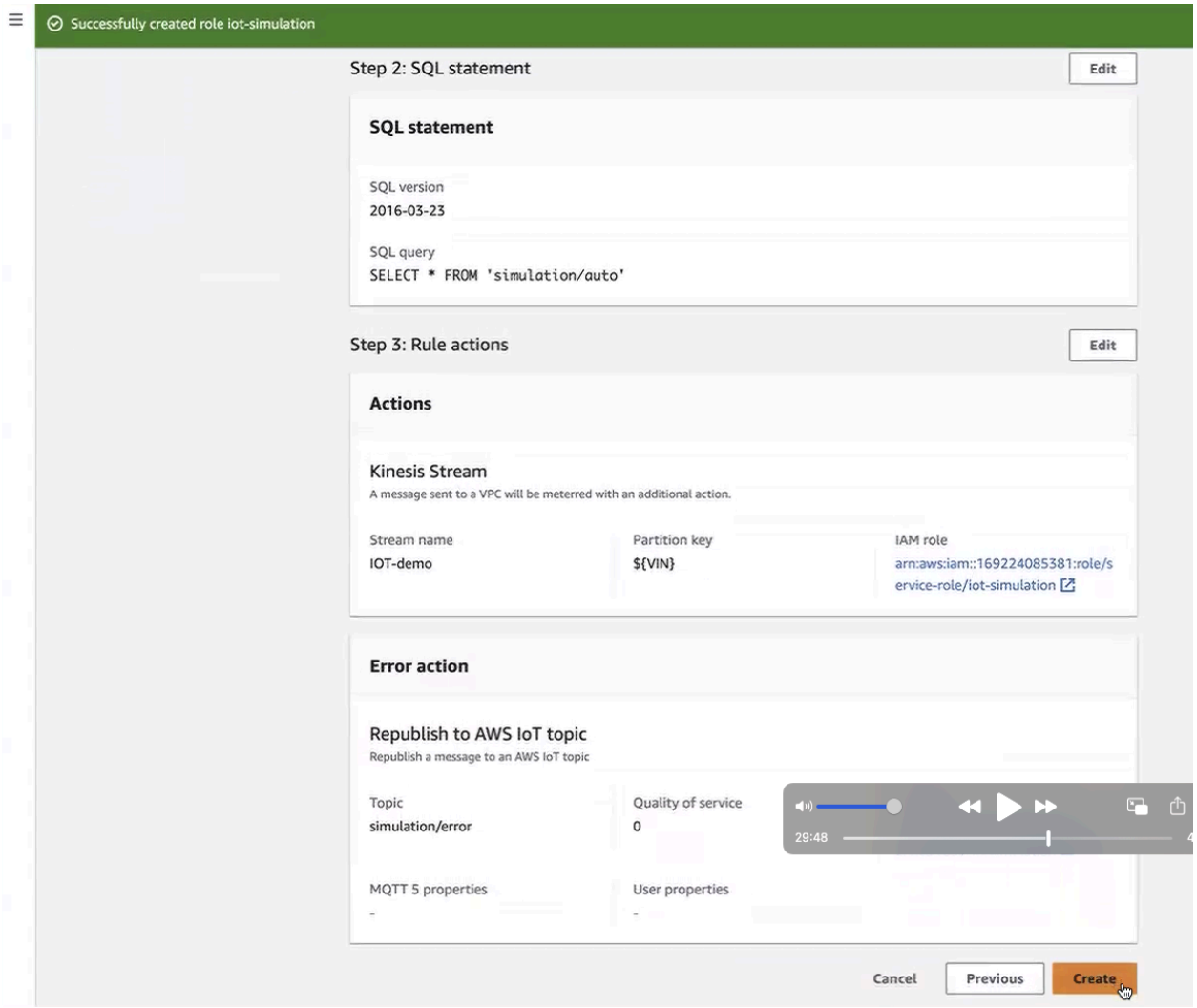
7. On this same page for the action we are working on, we will want to “Create new role”

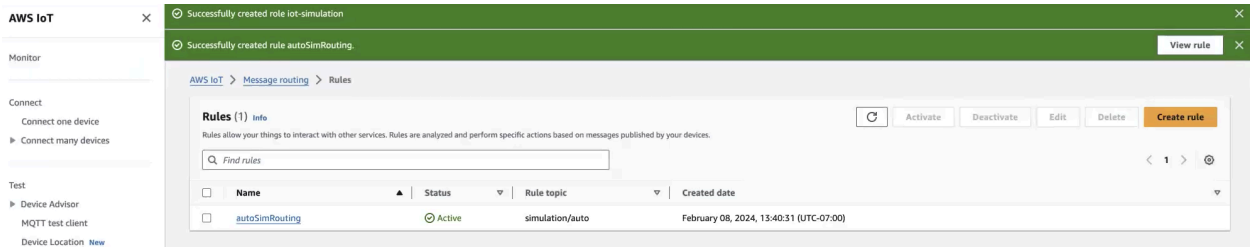


8. For the “Error action” section, select “Republish to AWS IoT topic” from the drop down, and select the same IAM role and click “Next”

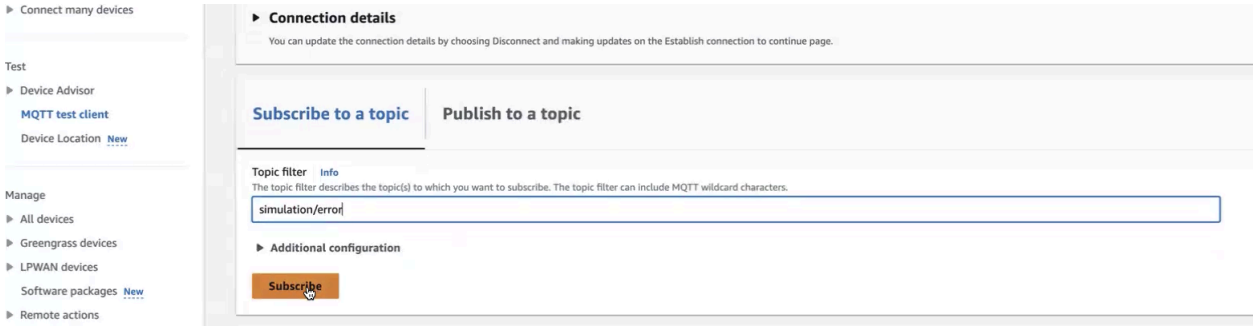

Error action - optional
You can optionally set an action that will be executed when something goes wrong with processing your rule. If two rule actions in the same rule fail, the error action receives one message that contains both errors.
Republish to AWS IoT topic
Republish a message to an AWS IoT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

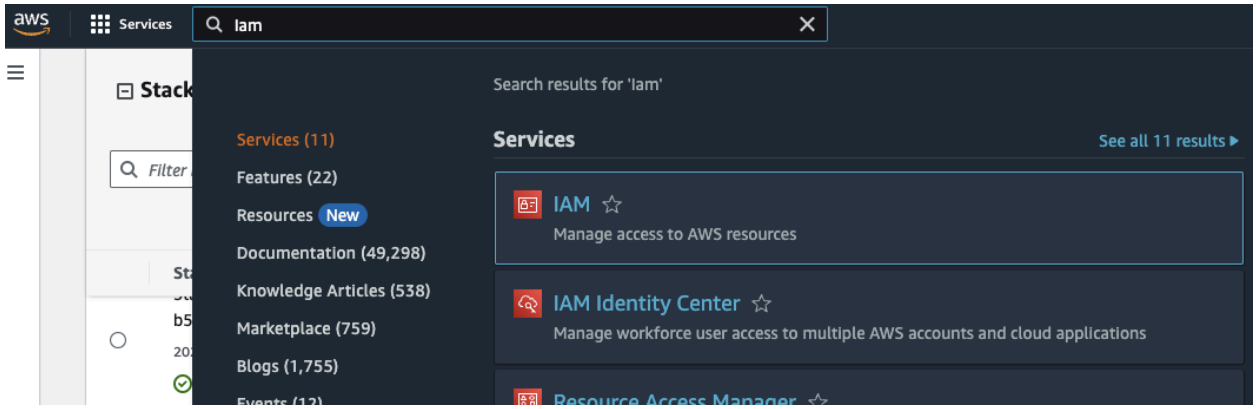




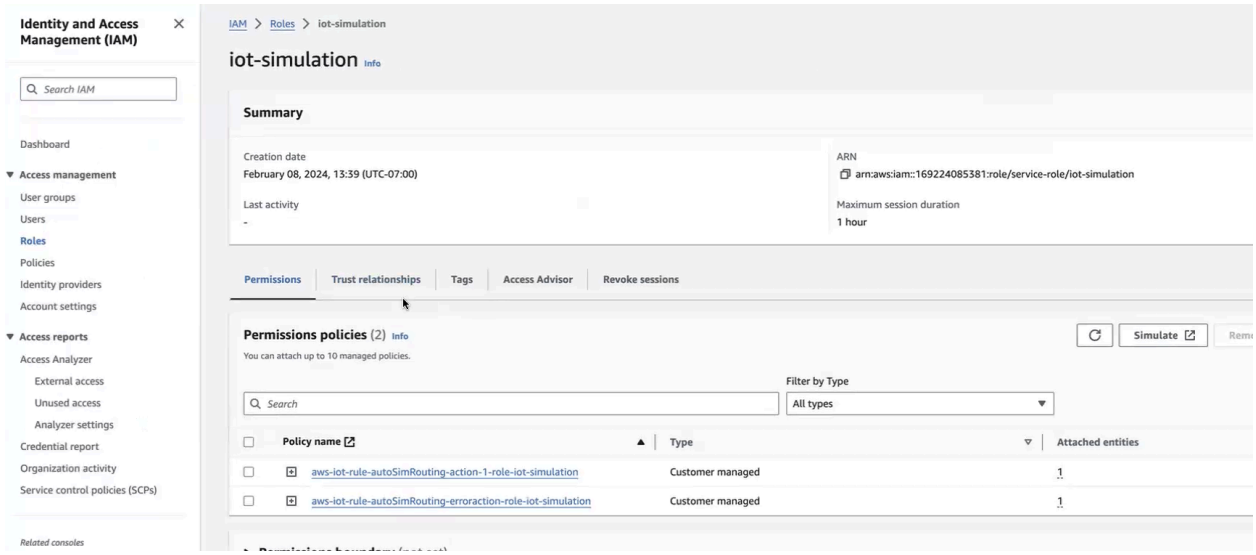
10. On the left menu click on “MQTT test client” and change the topic to “simulation/error” and click “subscribe”, until we add permissions in IAM our events will be going to the error topic



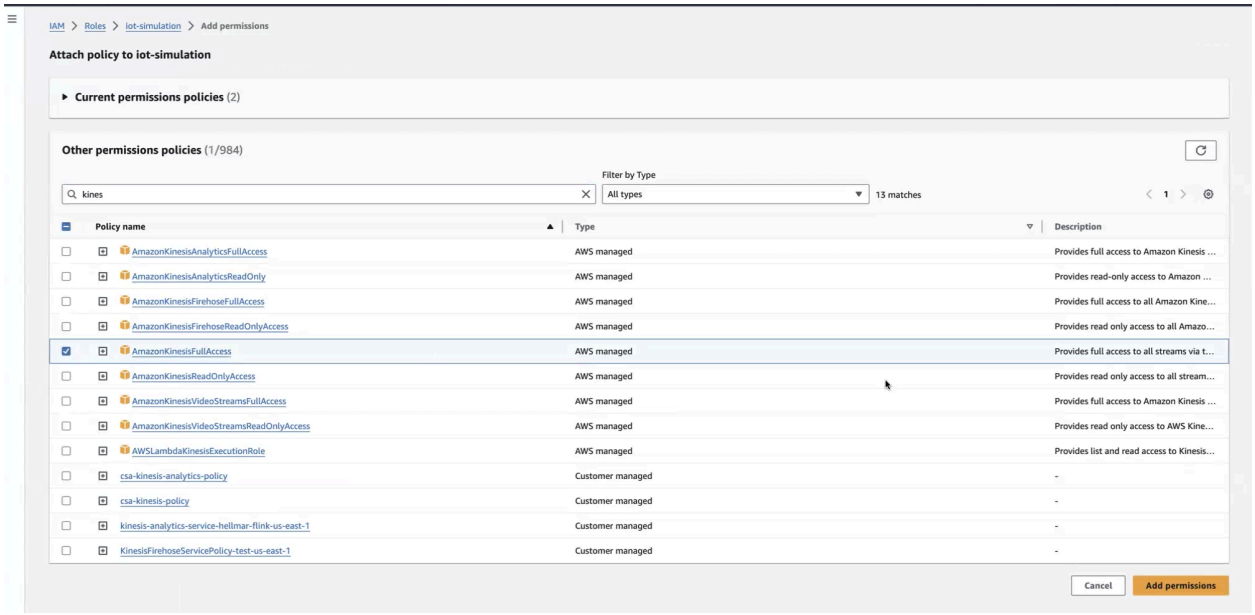
11. In another window, go to the IAM dashboard, and click “Roles”



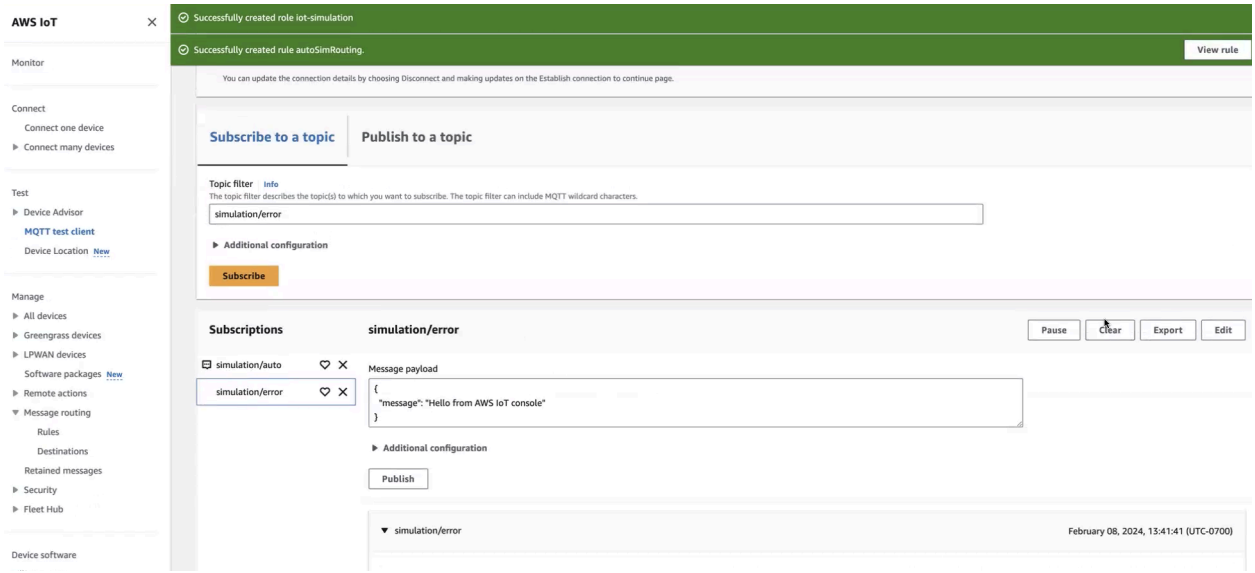
12. Find the newly created “iot-simulation” role, click the “Add permissions” button under the Permission policies section and click “attach policies”



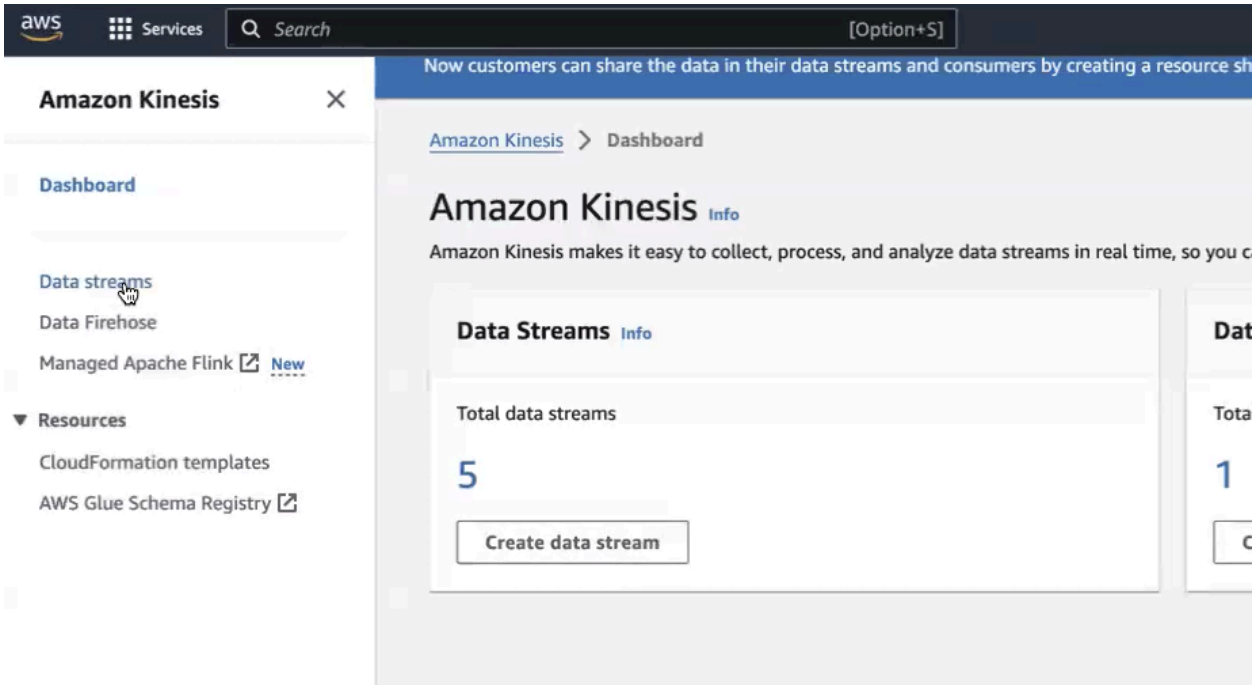
13. Add the “AmazonKinesisFullAccess” policy



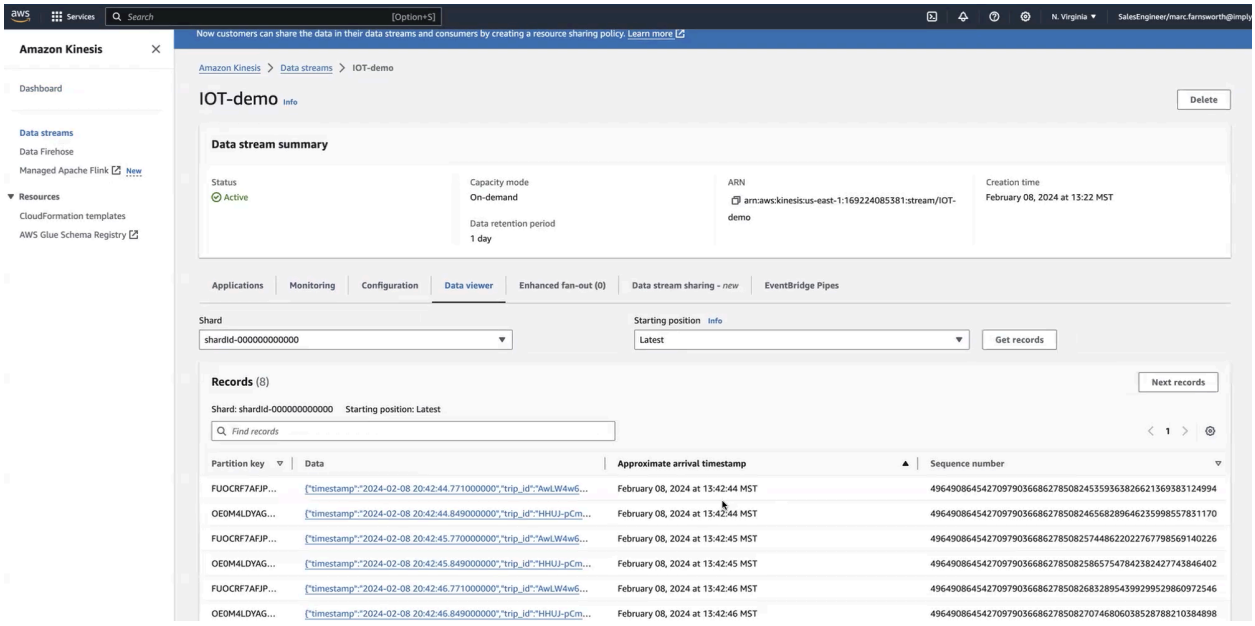
14. Navigate back to the MQTT test client, error events should no longer be streaming in



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

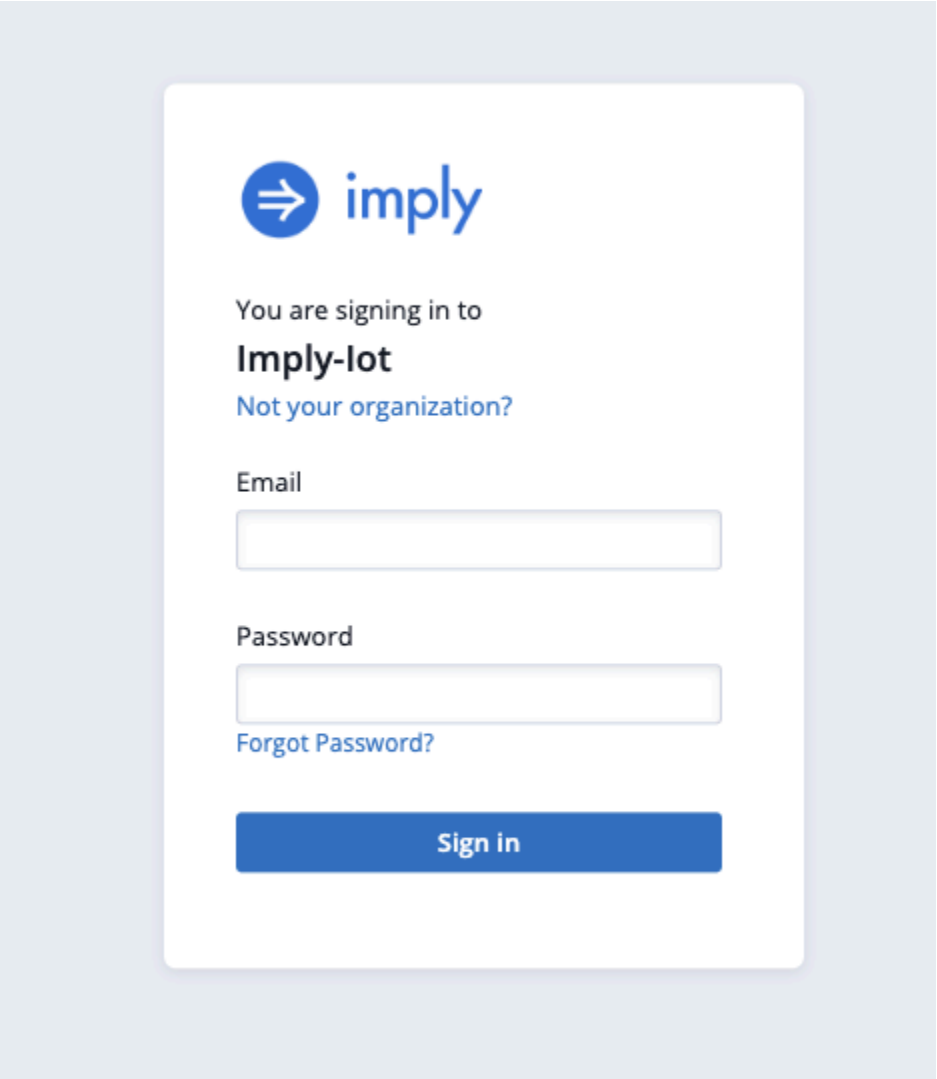


16. Click on “Data viewer” tab in the middle of the screen, under “Shard” select the first one, and then on the right side click “Next record” to validate events are now on the stream

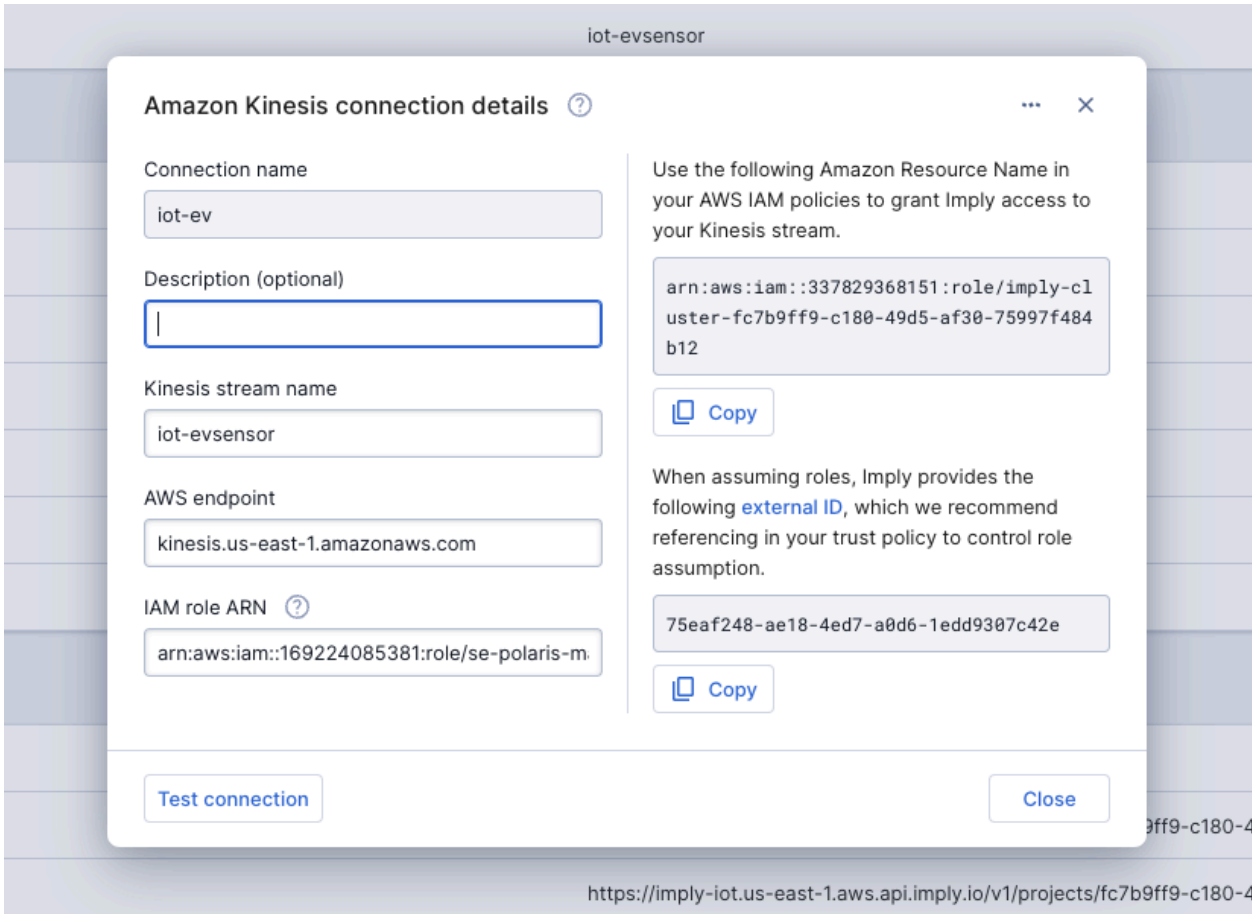


Step 5 - Connect Imply Polaris to Kinesis, and Create a Table

1. Sign into your Imply account

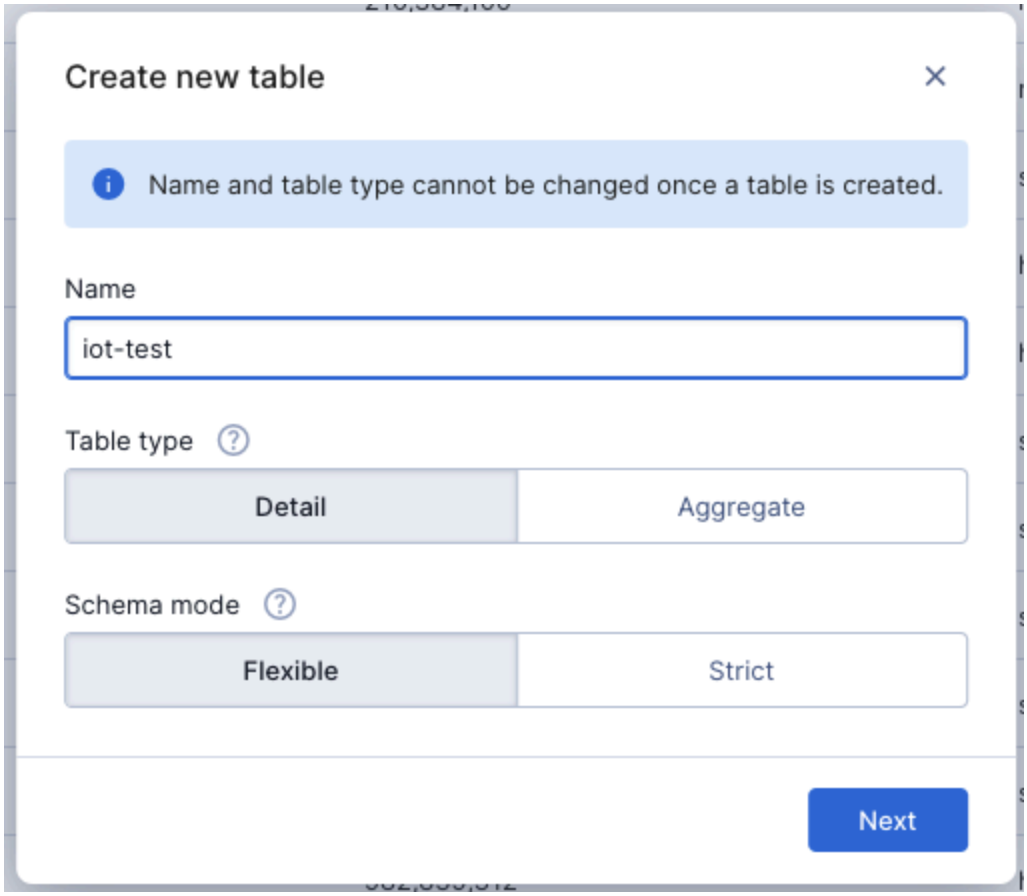


2. Click on “Sources” on the left nav, then at the top right click “Create source”. For a comprehensive guide, here are the docs https://docs.imply.io/polaris/ingestion-guide-kinesis

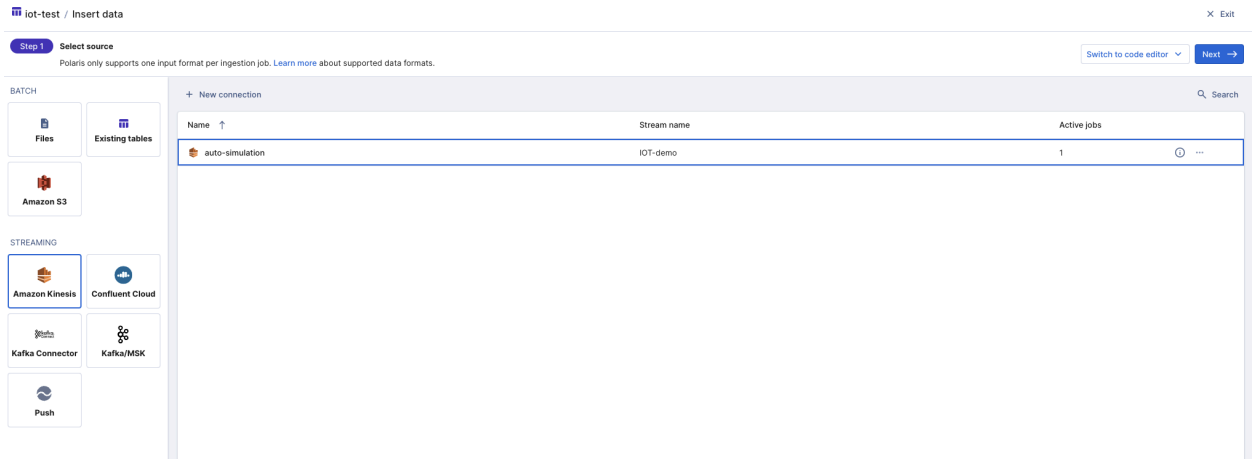


3. Open IAM in AWS and add ARN from the polaris provided ARN to the relevant kinesis role with trust policies

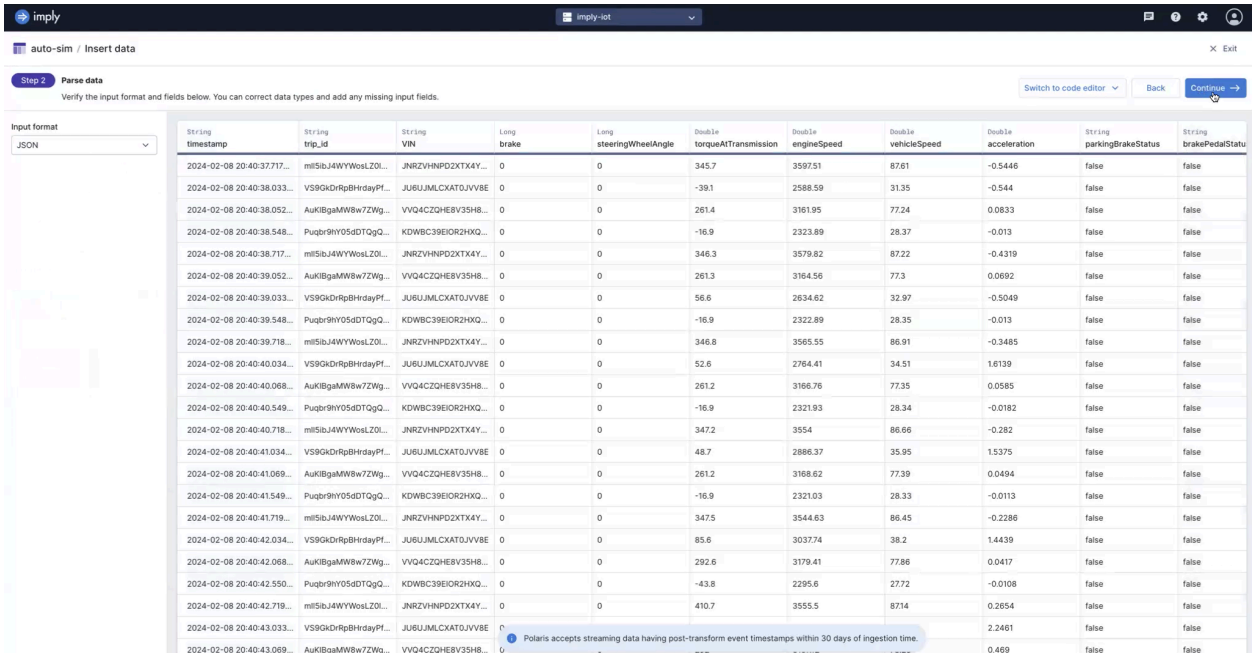
4. Create a table in Imply Polaris by clicking on the “Tables” tab and click “Create table” at the top right



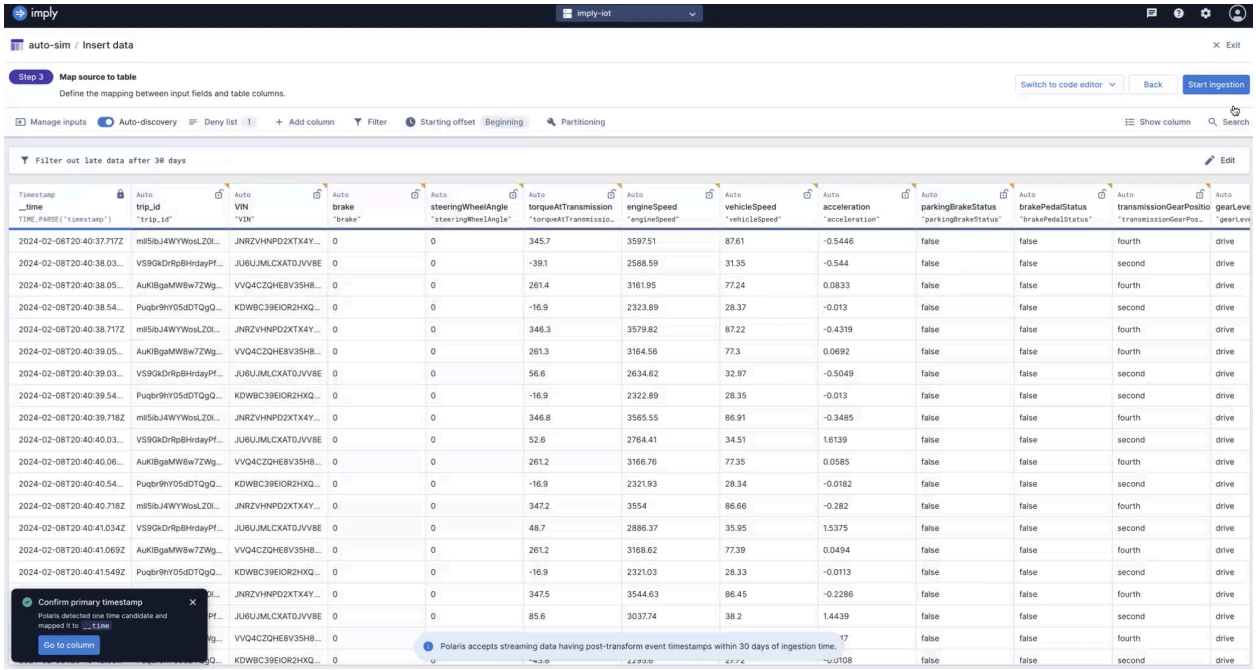
5. Click “Load data” and “Insert data”, select “Amazon Kinesis”, and select the “auto-simulation” stream then click “Next”



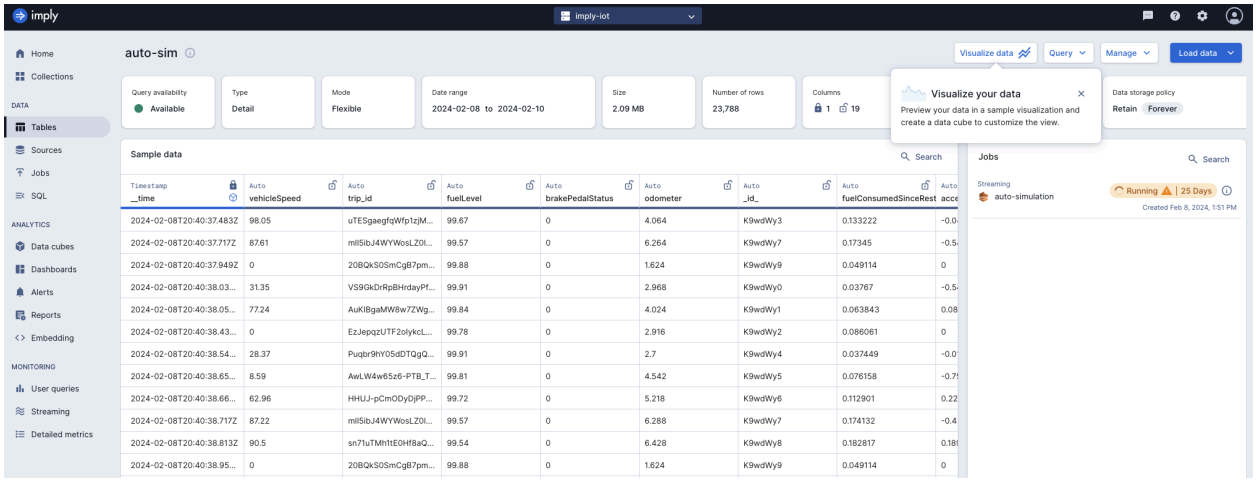
6. Polaris will auto identify they schema, click “Continue”



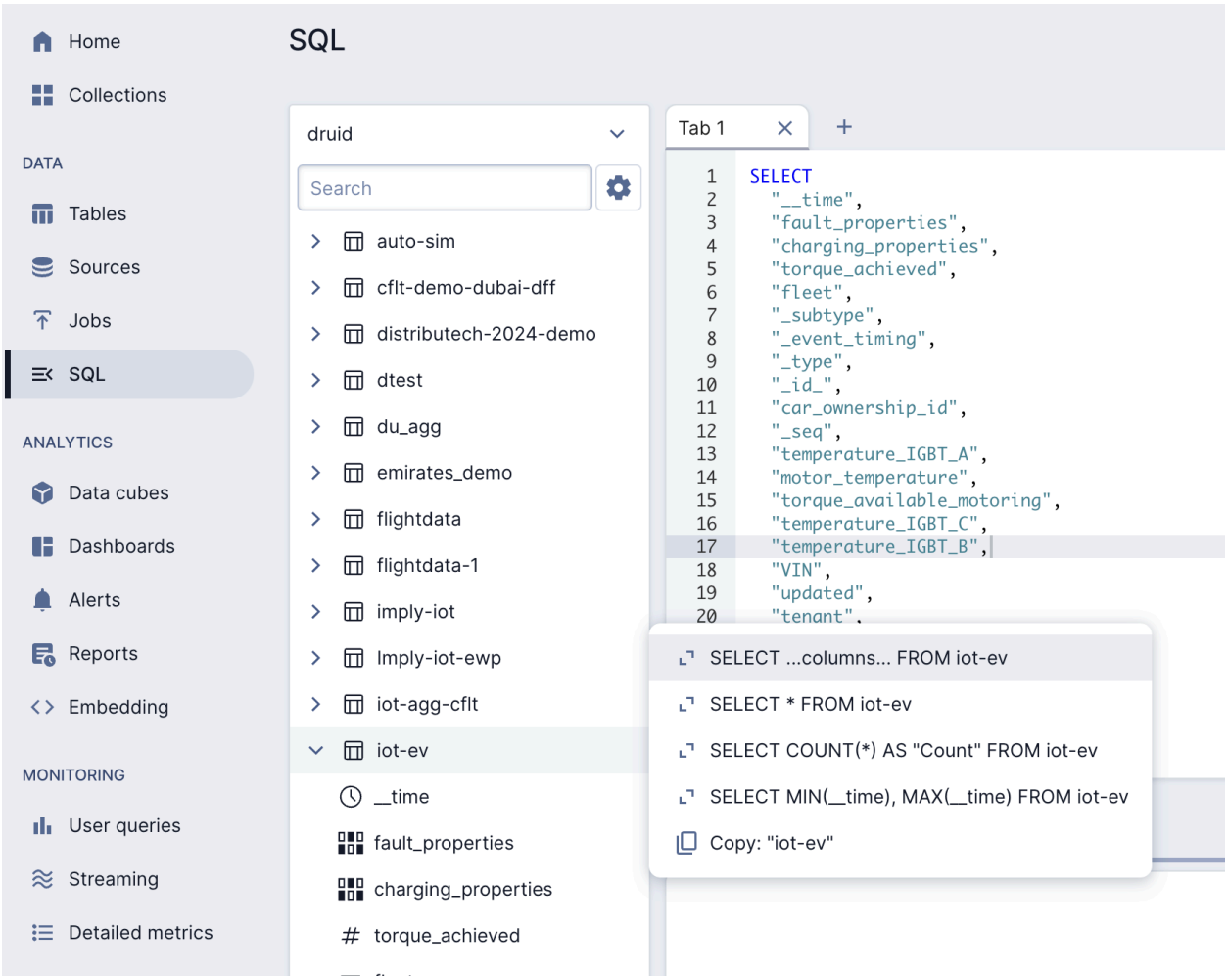
7. You can customize the table further, but for now in this test click “Start Ingestion”



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.



9. Lets query the table:
    a. From the home menu, on the left side click on “SQL”, click the newly created table and “Run” the query.

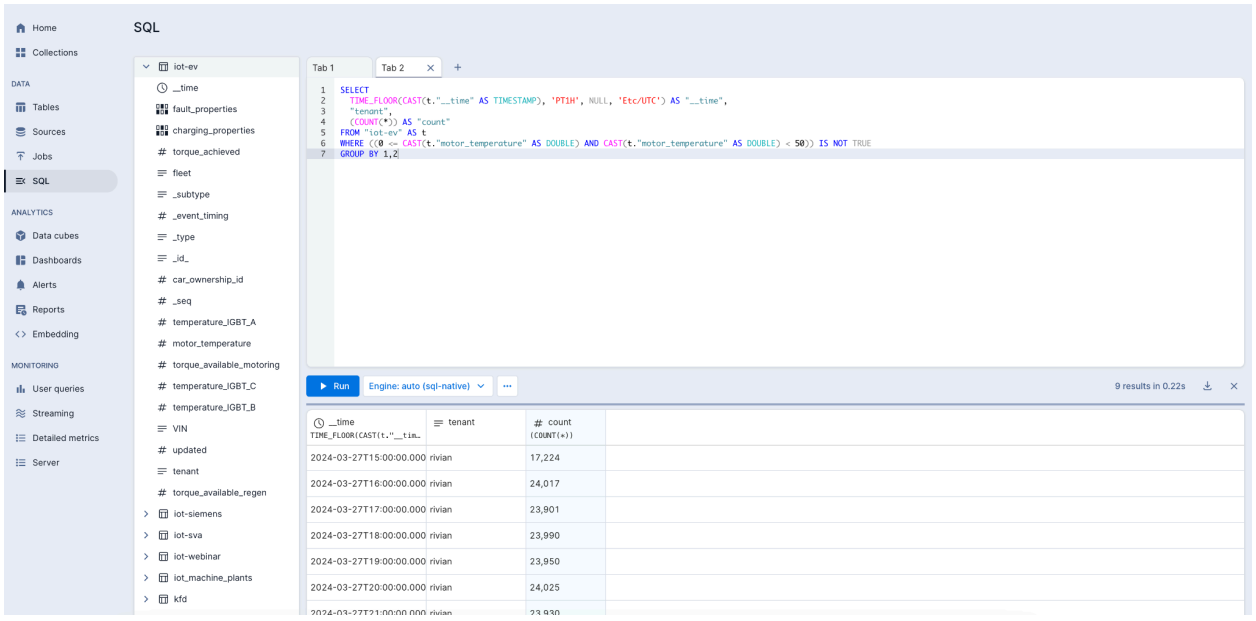


    b. Here are some example queries you can test in the SQL UI

Unset

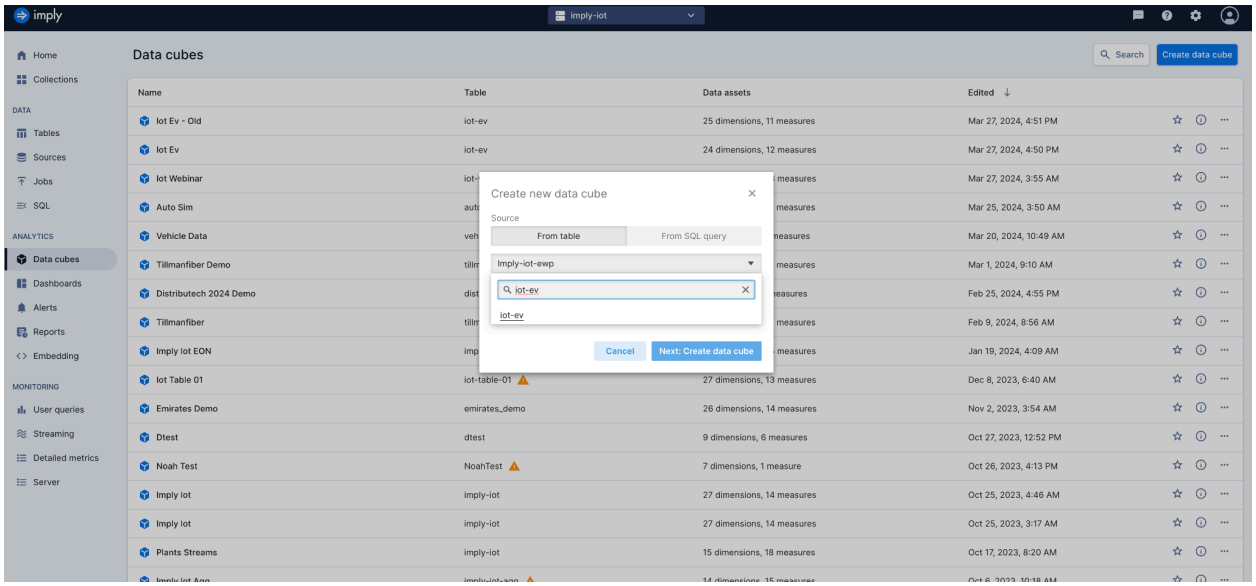
```
SELECT
  TIME_FLOOR(CAST(t."__time" AS TIMESTAMP), 'PT1H', NULL, 'Etc/UTC') AS
"__time",
  "tenant",
  (COUNT(*)) AS "count"
FROM "iot-ev" AS t
WHERE ((0 <= CAST(t."motor_temperature" AS DOUBLE) AND
CAST(t."motor_temperature" AS DOUBLE) < 50)) IS NOT TRUE
GROUP BY 1,2
```

```
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
```



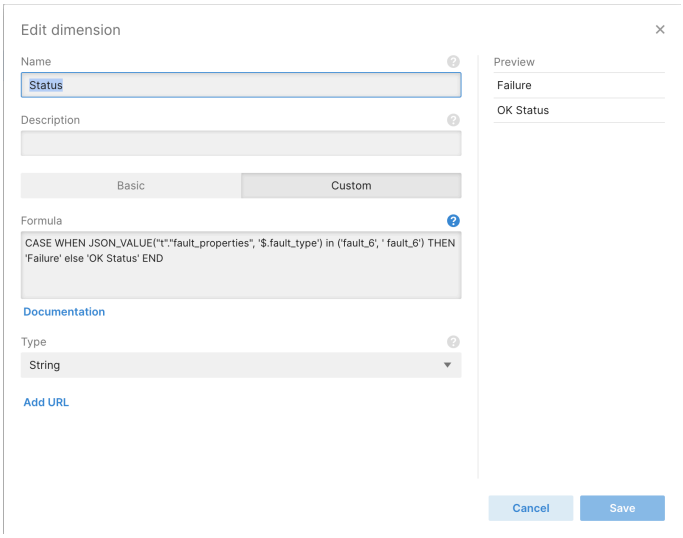
Step 6 - Real time Analytics & Dashboarding

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”



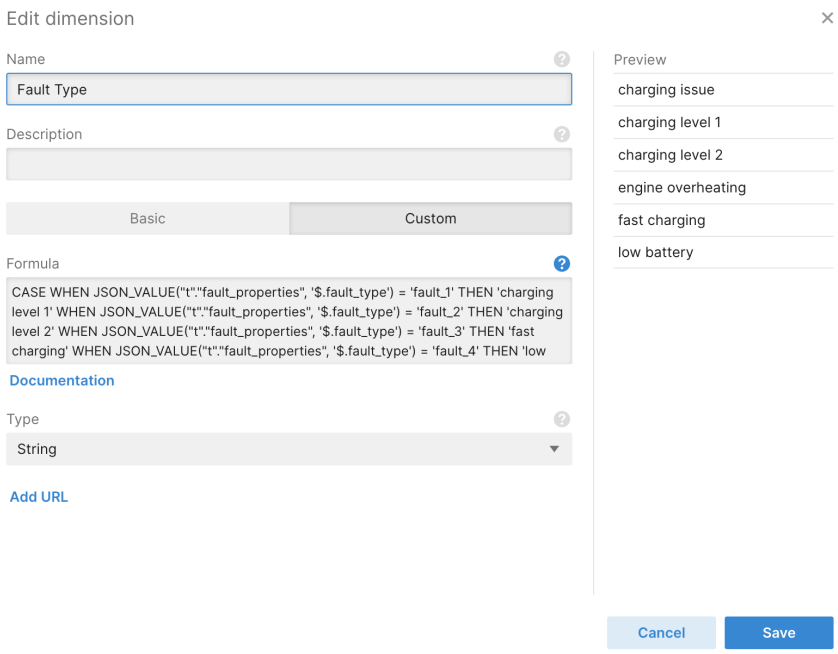
    b. From the drop down select your new table we created “iot-test”, and click “Next: Create data cube”. Leave the “Auto-fill dimensions and measures” checked
    c. Now let’s modify the **status** dimension and replace the default expression with this new one, it will help to simulate a failure status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

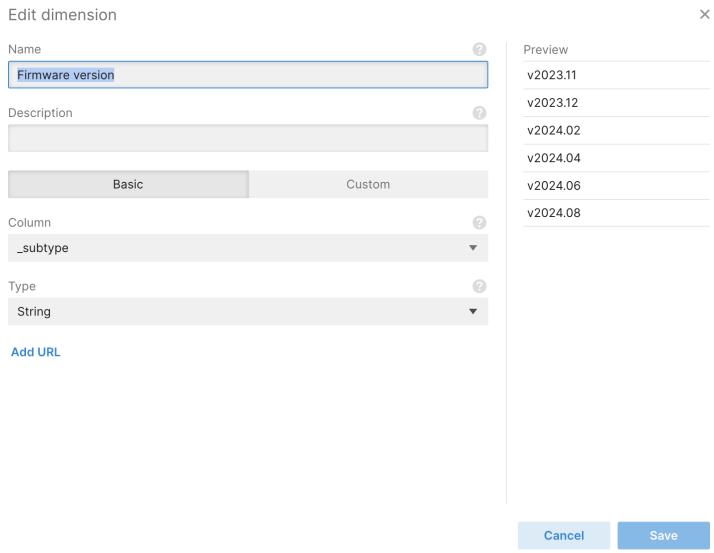


d. Now let's modify the **fault_type** dimension and replace the default expression with this new one in order to decode the fault type descriptions :

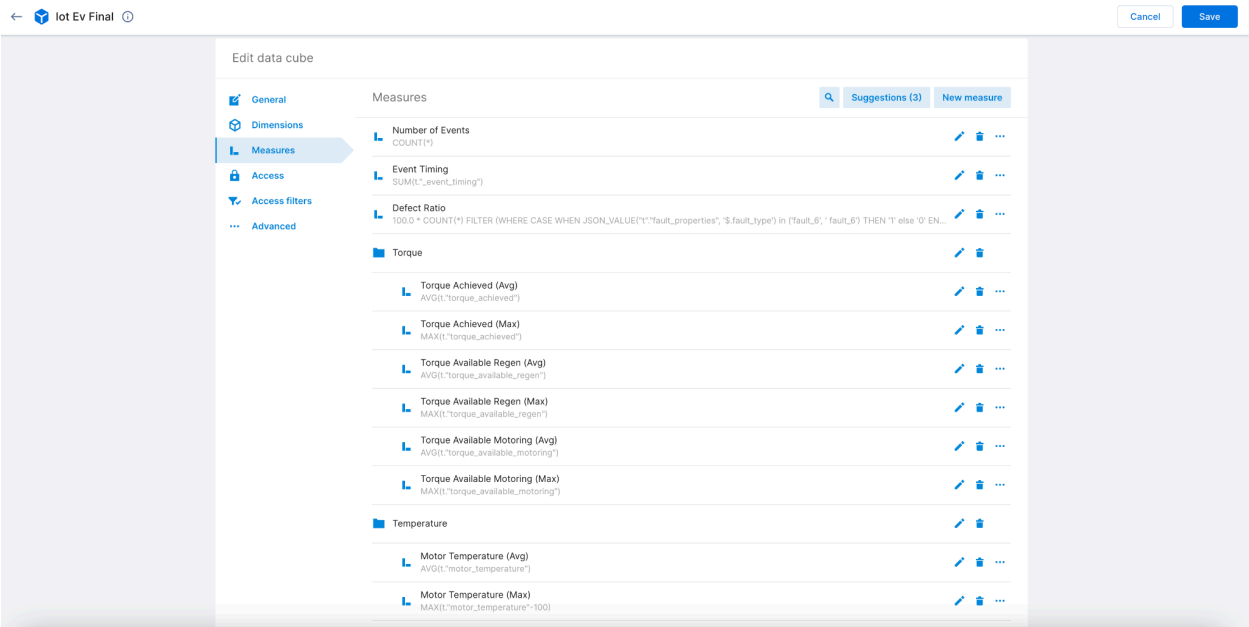
```
CASE WHEN JSON_VALUE("t"."fault_properties", '$.fault_type') = 'fault_1' THEN 'charging level 1' WHEN JSON_VALUE("t"."fault_properties", '$.fault_type') = 'fault_2' THEN 'charging level 2' WHEN JSON_VALUE("t"."fault_properties", '$.fault_type') = 'fault_3' THEN 'fast charging' WHEN JSON_VALUE("t"."fault_properties", '$.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
```



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



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



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

Torque Achieved (Avg)
AVG(t."torque_achieved")

Torque Achieved (Max)
MAX(t."torque_achieved")

Torque Available Regen (Avg)
AVG(t."torque_available_regen")

Torque Available Regen (Max)
MAX(t."torque_available_regen")

Torque Available Motoring (Avg)
AVG(t."torque_available_motoring")

Torque Available Motoring (Max)
MAX(t."torque_available_motoring")

Motor Temperature (Avg)
AVG(t."motor_temperature")

Motor Temperature (Max)
MAX(t."motor_temperature"-100)

Temperature IGBT A (Avg)
AVG(t."temperature_IGBT_A")

Temperature IGBT A (Max)
MAX(t."temperature_IGBT_A"-100)

Temperature IGBT B (Avg)
AVG(t."temperature_IGBT_B")

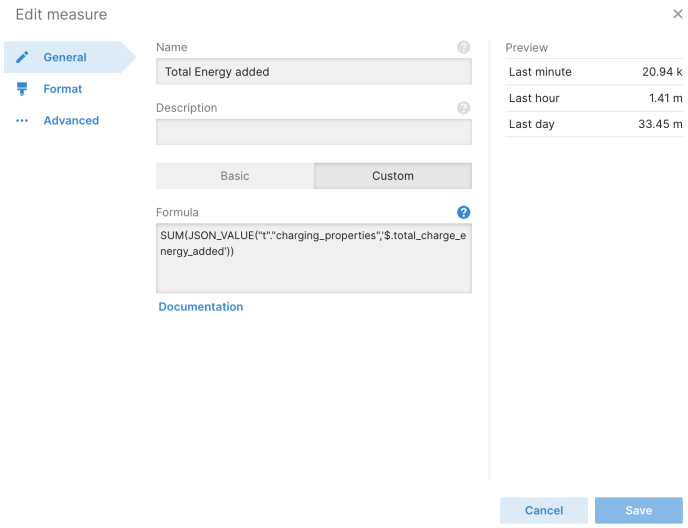
Temperature IGBT B (Max)
MAX(t."temperature_IGBT_B"-100)

Temperature IGBT C (Avg)
AVG(t."temperature_IGBT_C")

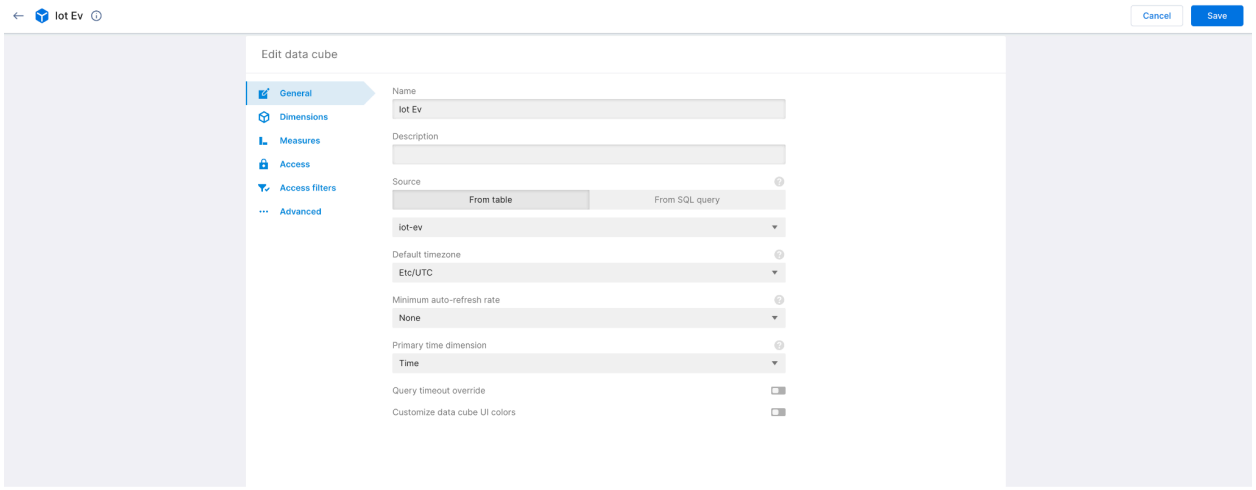
Temperature IGBT C (Max)
MAX(t."temperature_IGBT_C"-100)

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

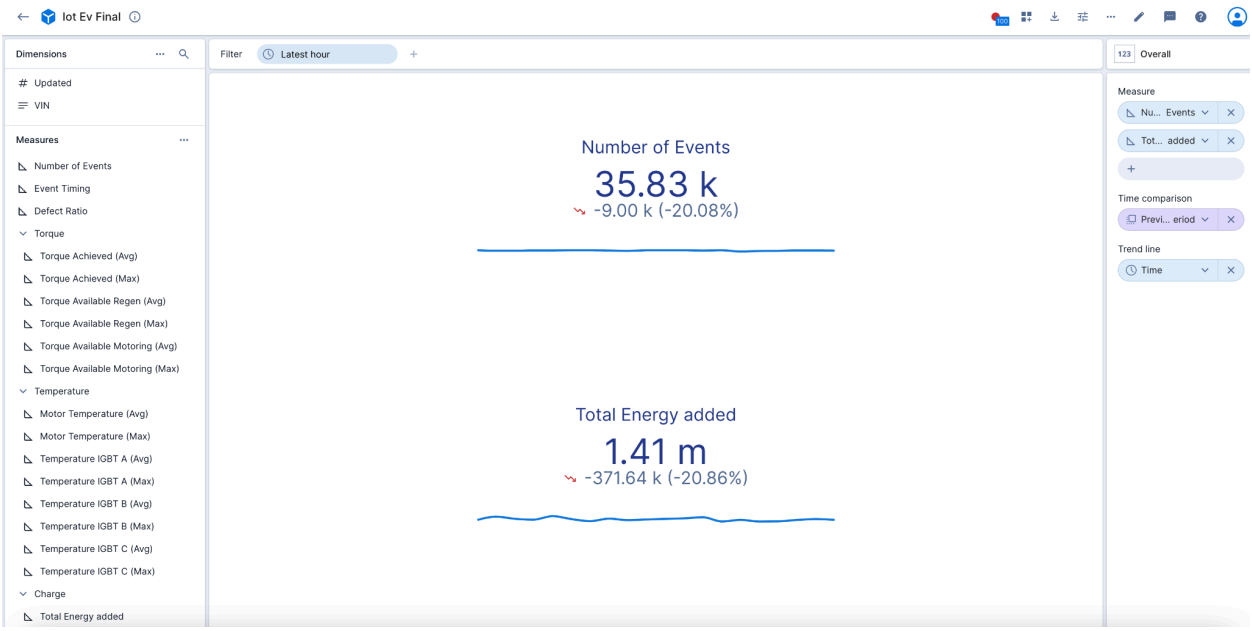
```
SUM(JSON_VALUE("t"."charging_properties",'$.total_charge_energy_added'))
```



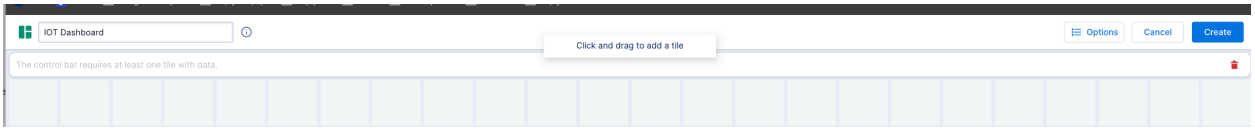
h. Give your datacube a name. And click “Save” :



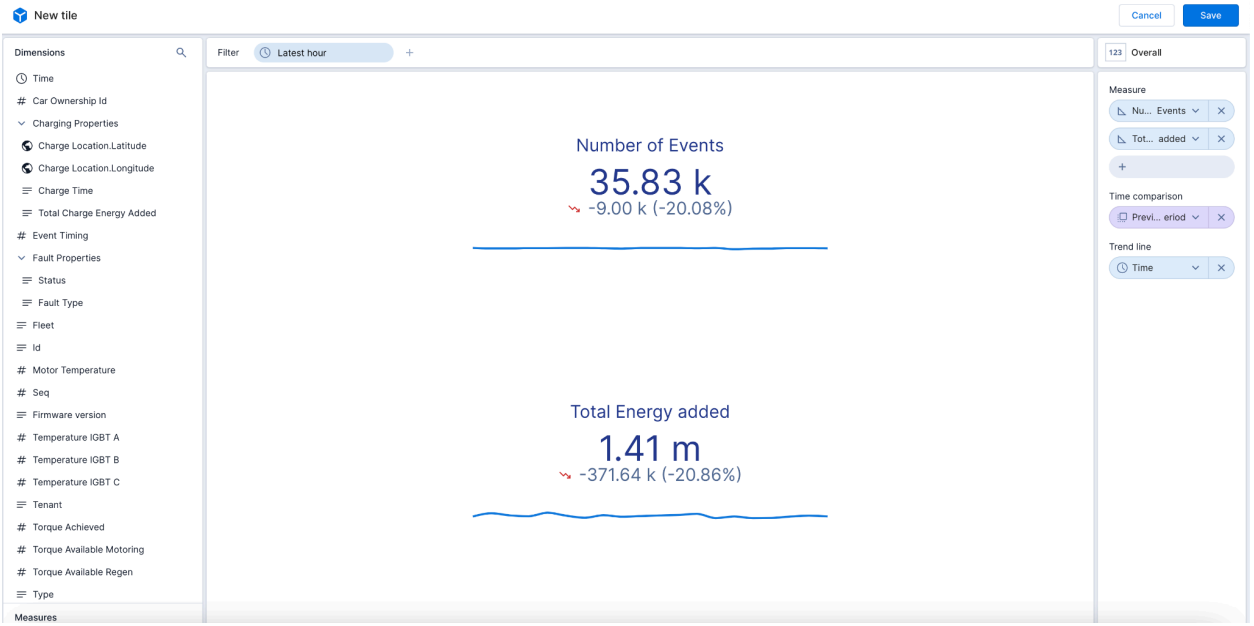
i. Now you can use the UI to drag and drop and analyze the data with existing dimensions and measures, for example, drag “Total Energy added” onto the measure lists. You can do this with multiple measures and change the measures according to your need. From here you can click  at the top right, to add 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.



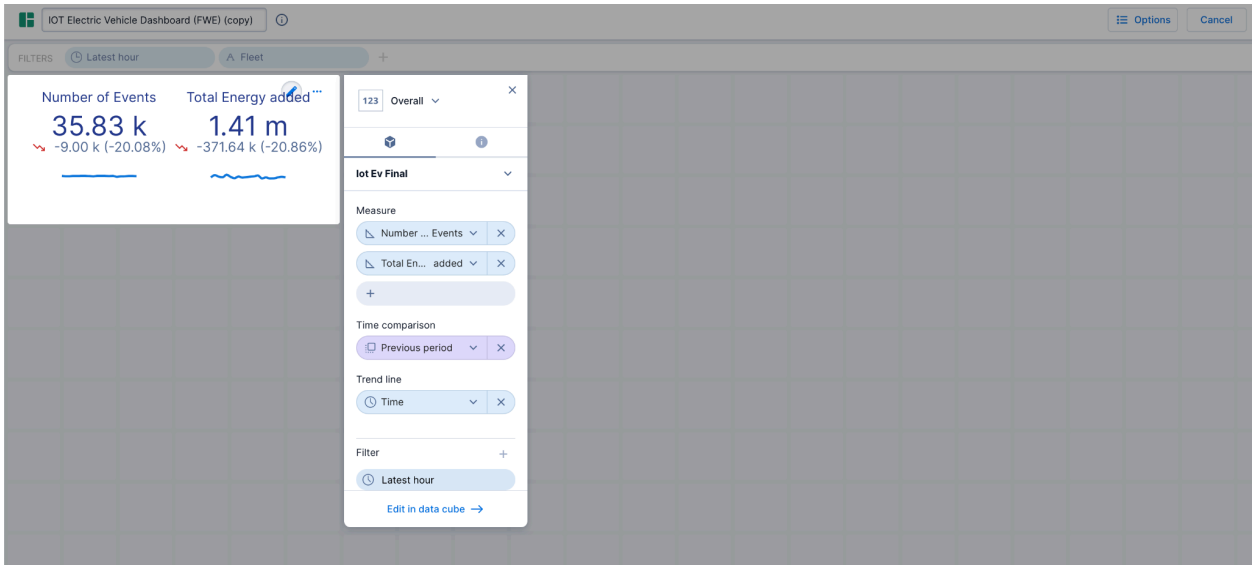
2. Creating a dashboard from a datacube is easy:
    a. Navigate back to the home screen and click “Dashboards” and “Create dashboard” at the top. Name the dashboard as “IOT Dashboard”



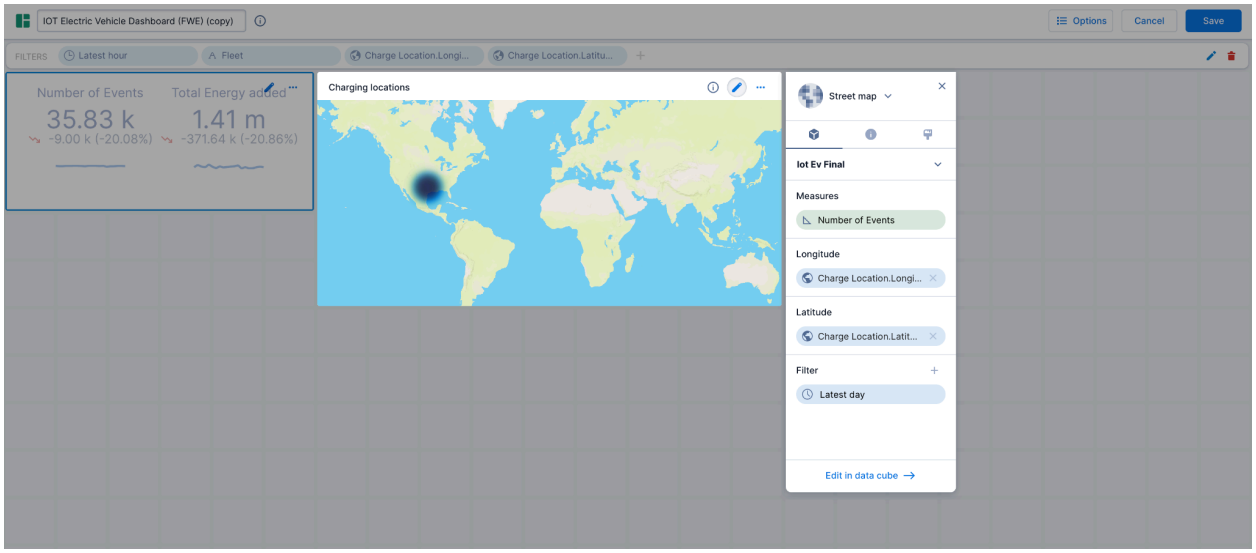
    b. Click and select an area on the dashboard, in the drop down select the cube you just created, it may be selected by default if you only have one cube in your polaris project. For chart type, select “Overall”, and click “Edit in data cube”, just create the same Overall visualization as previous step,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



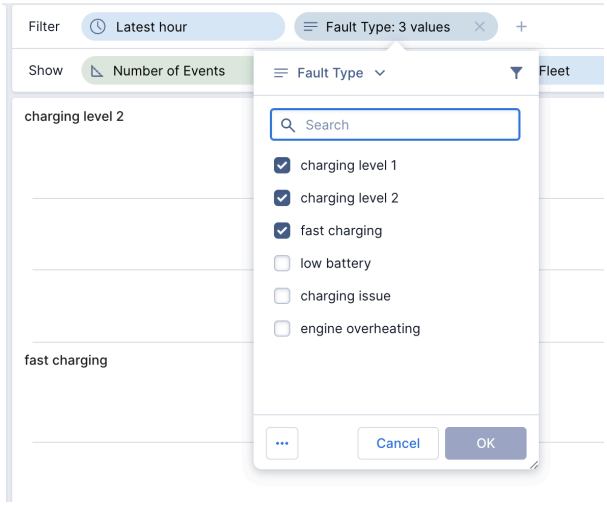
Visualization is added to the dashboard :



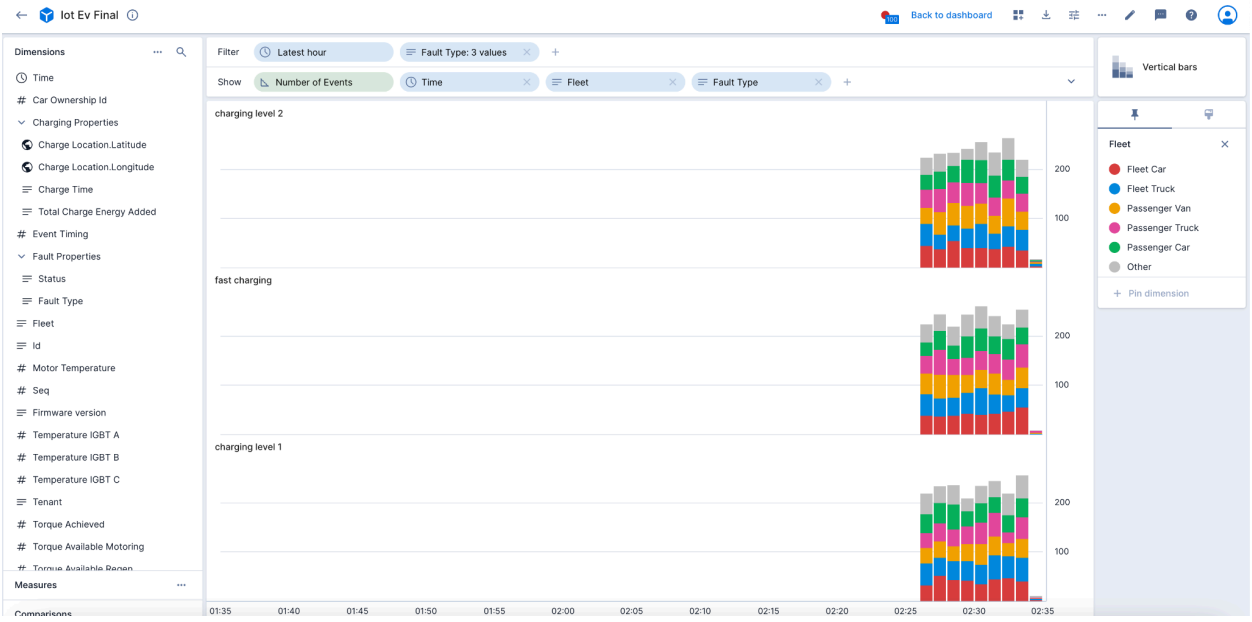
c. Now, let's create a new chart against the same datacube, select the **street map** visualization, it will automatically propose to use the 2 existing geo dimensions **LAT** and **LON** :



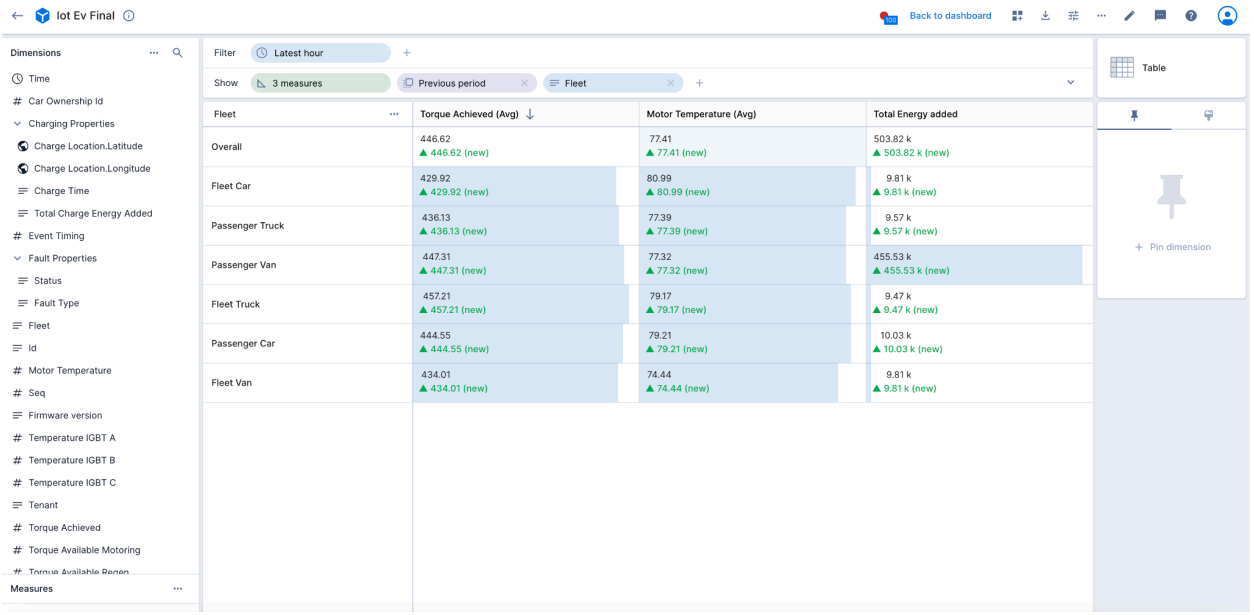
d. Now, let's create a new chart on our dashboard against the same datacube to monitor the charging status per fleet type, select vertical bars chart type, then add **time**, **fleet** and **fault type**, also add a filter on **fault type** restricted to the 3 charging types below :



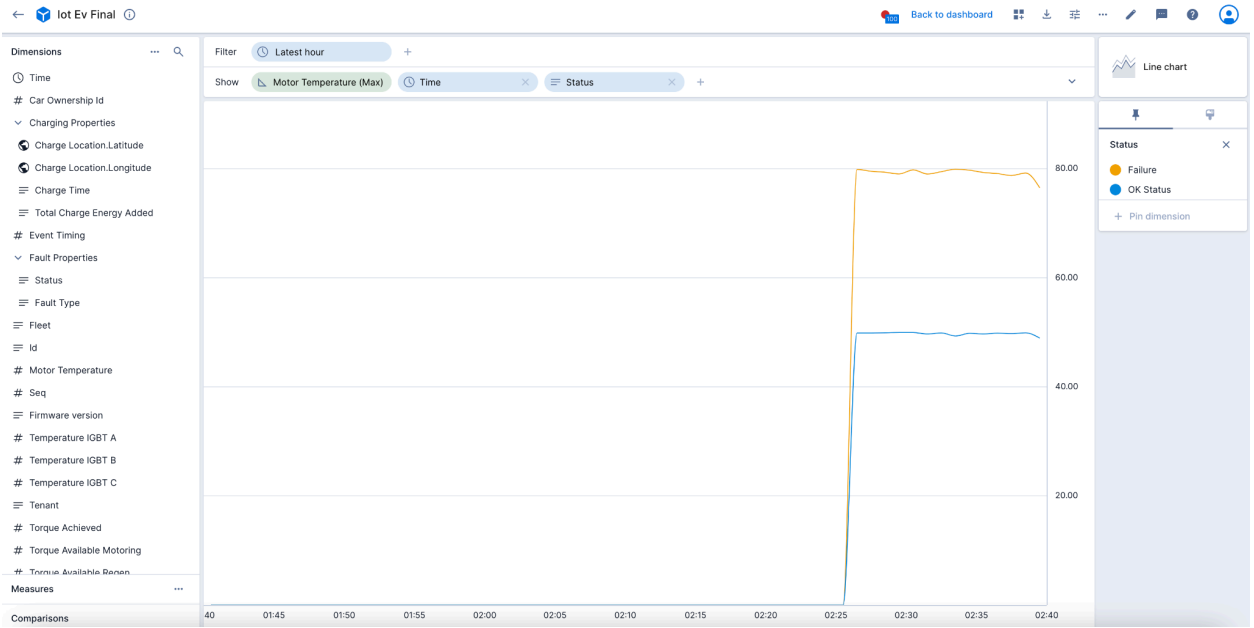
Your report should look like this before you add it to the dashboard :



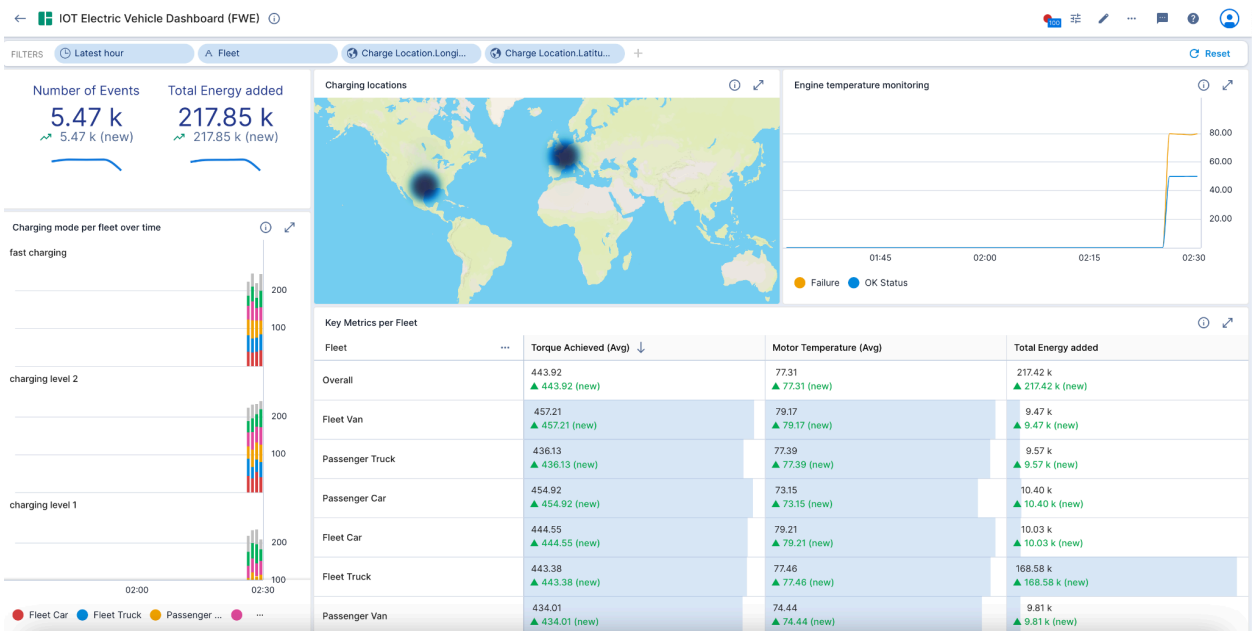
e. Now let's add a new chart to monitor Avg Torque, Avg Motor Temperature and Total energy added by selecting a Table visualization, adding these 3 measures and adding the previous period for comparison, report should look like this before you add it to the dashboard :



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 your 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 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