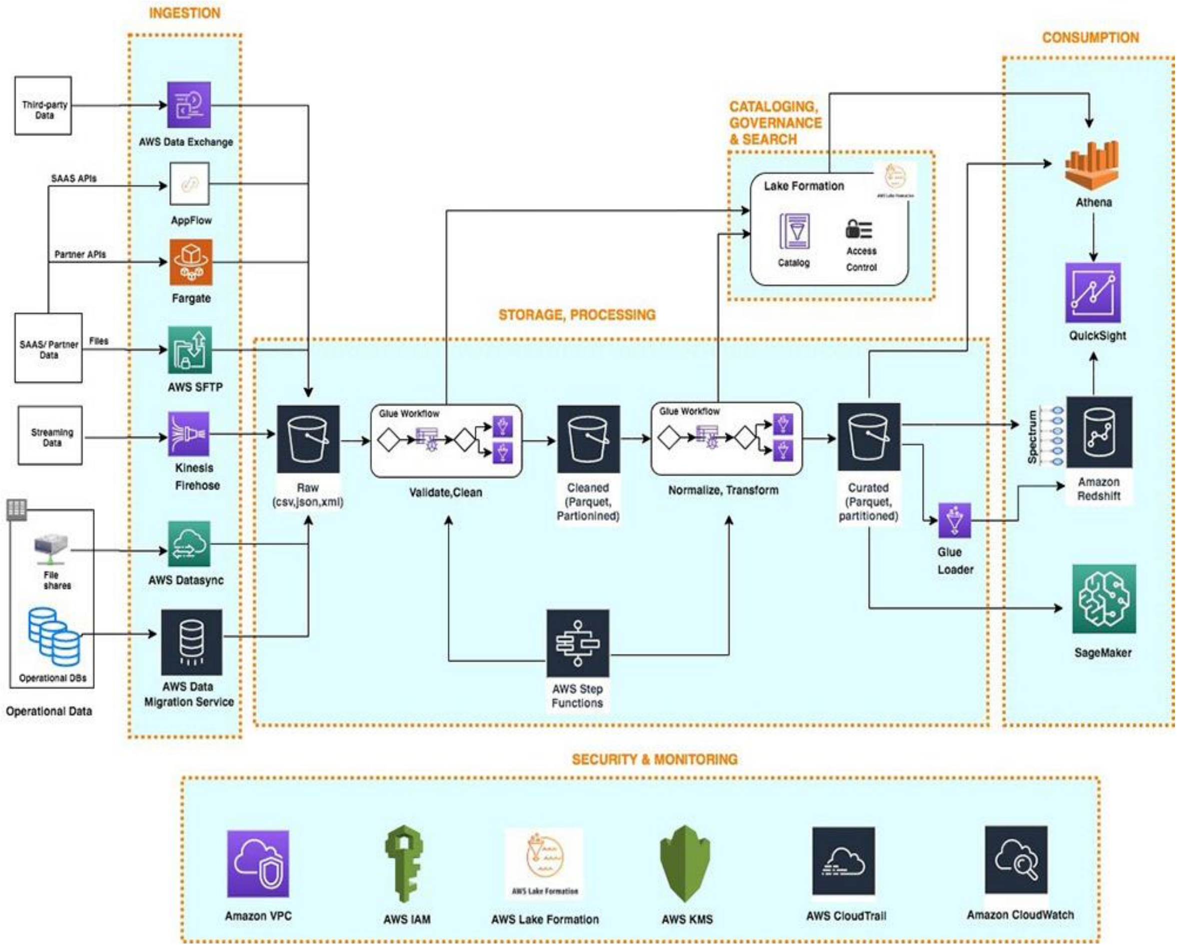
CREATE A SERVERLESS IOT DATA PROCESSING

Onboarding new data or building new analytics pipelines in traditional analytics architectures typically requires extensive coordination across business, data engineering, and data science and analytics teams to first negotiate requirements, schema, infrastructure capacity needs, and workload management.

For many use cases today however, business users, data scientists, and analysts are demanding easy, frictionless, self-service options to build end-to-end data pipelines because it’s hard and inefficient to predefine constantly changing schemas and spend time negotiating capacity slots on shared infrastructure. The exploratory nature of machine learning (ML) and many analytics tasks means you need to rapidly ingest new datasets and clean, normalize, and feature engineer them without worrying about operational overhead when you must think about the infrastructure that runs data pipelines.

Serverless data lake centric analytics architecture:



To compose the layers described in our logical architecture, we introduced a reference architecture that uses serverless and managed services.

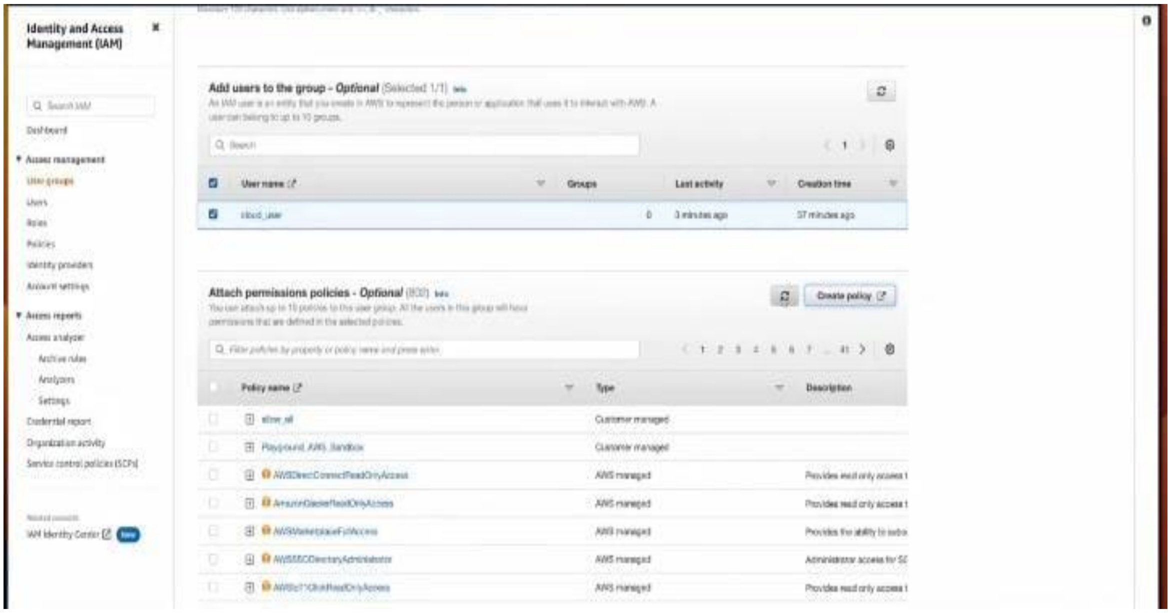
It provides the following key benefit:

- Easy configuration-driven use.

- Freedom from infrastructure management.
- Pay-per-use pricing model.

INGESTION LAYER:

The ingestion layers in our serverless architecture are composed of a set of purpose-built services to enable data ingestion from a variety of sources.



Add users to the group - Optional (Selected 1/1) Info

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS. A user can belong to up to 10 groups.

☑	User name	Groups	Last activity	Creation time
☑	cloud_user	0	3 minutes ago	37 minutes ago

Attach permissions policies - Optional ([illegible]) Info

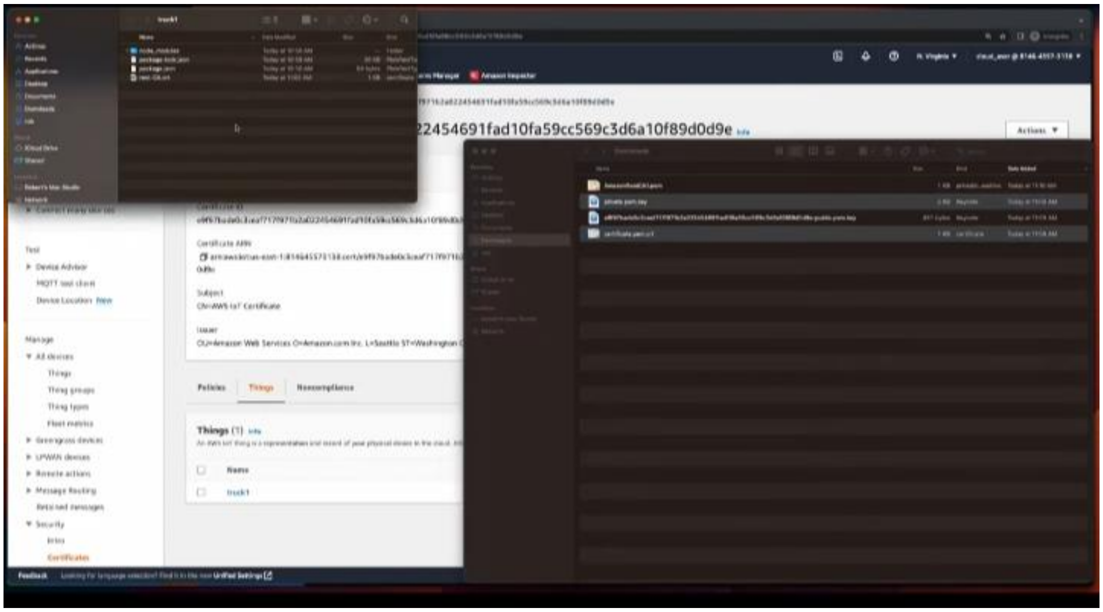
You can attach up to 10 policies to this user group. All the users in this group will have permissions that are defined in the selected policies.

Create policy

□	Policy name	Type	Description
□	allow_all	Customer managed	
□	Playground_AWS_Sandbox	Customer managed	
□	AWSDirectConnectReadOnlyAccess	AWS managed	Provides read only access t
□	AmazonGlacierReadOnlyAccess	AWS managed	Provides read only access t
□	AWSMarketplaceFullAccess	AWS managed	Provides the ability to subs
□	AWSSSODirectoryAdministrator	AWS managed	Administrator access for SS
□	AWSIoT1ClickReadOnlyAccess	AWS managed	Provides read only access t

STORAGE LAYER:

Data of any structure and any format can be stored as S3 objects without needing to predefine any schema. This enables services in the ingestion layer to quickly land a variety of source data into the data lake in its original source format.

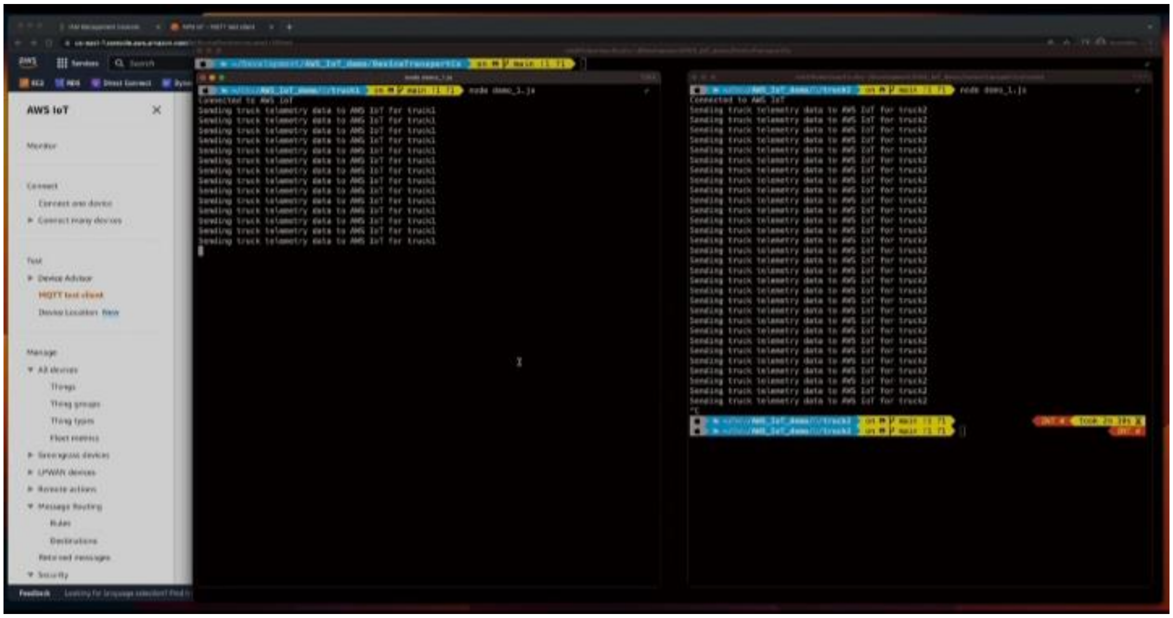


Subject
CN=AWS IoT Certificate

Issuer
OU=Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington

Cataloging and search layer:

A data lake typically hosts a large number of datasets, and many of these datasets have evolving schema and new data partitions. A central Data Catalog that manages meta data for all the datasets in the data lake is crucial to enabling self-service discovery of data in the data lake. Additionally, separating metadata from data into a central schema enables schema-on-read for the processing and consumption layer components.



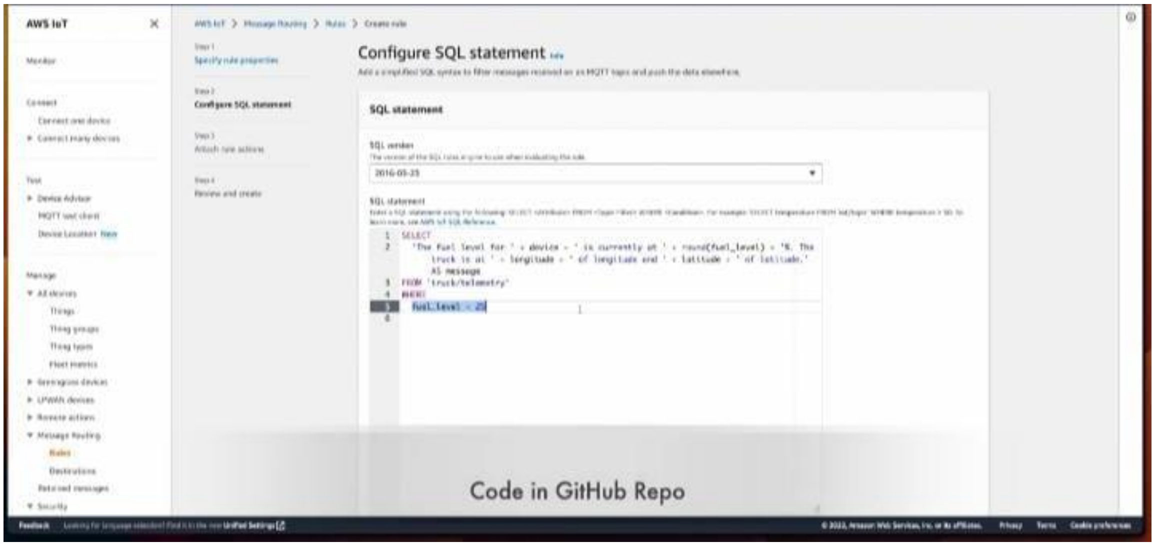
```
node demo_1.js
Connected to AWS IoT
Sending truck telemetry data to AWS IoT for truck1
Sending truck telemetry data to AWS IoT for truck1
Sending truck telemetry data to AWS IoT for truck1
Sending truck telemetry data to AWS IoT for truck1
Sending truck telemetry data to AWS IoT for truck1
Sending truck telemetry data to AWS IoT for truck1
Sending truck telemetry data to AWS IoT for truck1
Sending truck telemetry data to AWS IoT for truck1
Sending truck telemetry data to AWS IoT for truck1
Sending truck telemetry data to AWS IoT for truck1
Sending truck telemetry data to AWS IoT for truck1
Sending truck telemetry data to AWS IoT for truck1
Sending truck telemetry data to AWS IoT for truck1
Sending truck telemetry data to AWS IoT for truck1
```

```
node demo_1.js
Connected to AWS IoT
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
Sending truck telemetry data to AWS IoT for truck2
^C
```

Processing layer:

The processing layer in our architecture is composed of two types of components:

- Components used to create multi-step data processing pipelines
- Components to orchestrate data processing pipelines on schedule or in response to event triggers (such as ingestion of new data into the landing zone)



Configure SQL statement

SQL statement

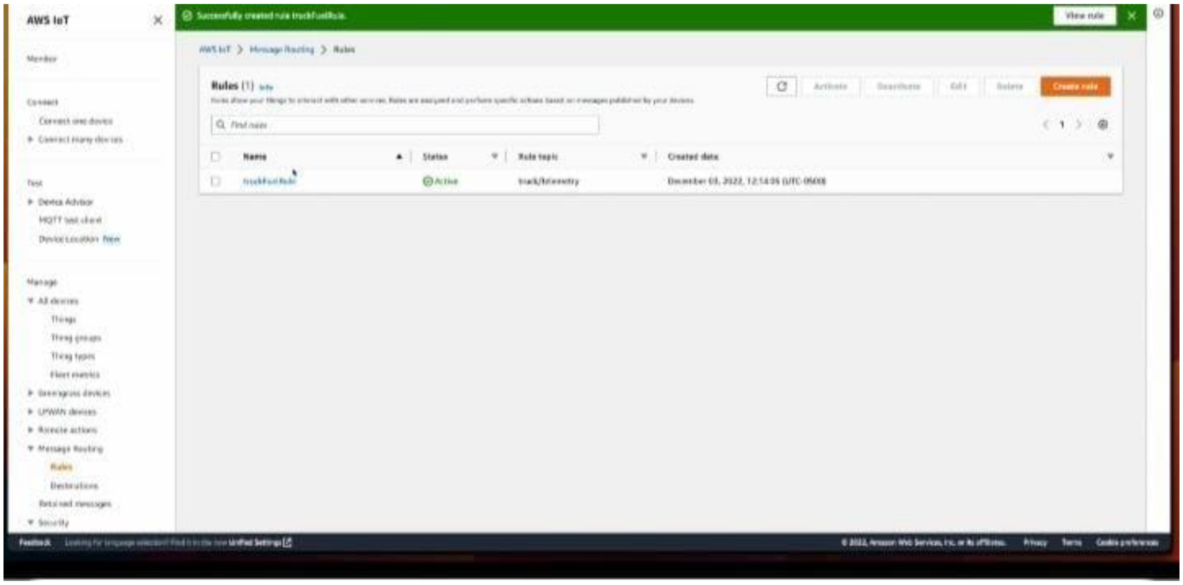
SQL version: 2016-03-23

```
SELECT
  'The fuel level for ' + device + ' is currently at ' + round(fuel_level) + '%. The
      truck is at ' + longitude + ' of longitude and ' + latitude + ' of latitude.'
      AS message
FROM 'truck/telemetry'
WHERE
  fuel_level < 25
```

Code in GitHub Repo

Consumption layer:

The consumption layer in our architecture is composed using fully managed, purpose-built, analytics services that enable interactive SQL, BI dashboarding, batch processing, and ML.



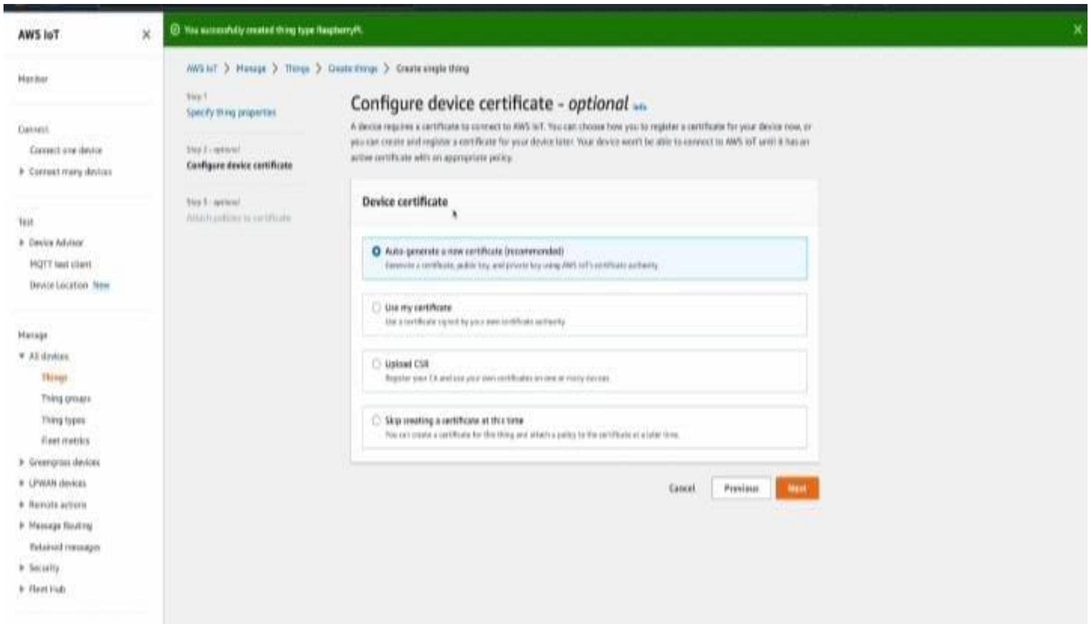
Successfully created rule truckFuelRule.

Rules (1)

Name	Status	Rule topic	Created date
truckFuelRule	Active	truck/telemetry	December 03, 2022, 12:14:05 (UTC-0500)

Security and governance layer:

Components across all layers of our architecture protect data, identities, and processing resources by natively using the following capabilities provided by the security and governance layer.



You successfully created thing type RaspberryPi.

Configure device certificate - *optional* Info

A device requires a certificate to connect to AWS IoT. You can choose how you to register a certificate for your device now, or you can create and register a certificate for your device later. Your device won't be able to connect to AWS IoT until it has an active certificate with an appropriate policy.

Device certificate

● Auto-generate a new certificate (recommended)
Generate a certificate, public key, and private key using AWS IoT's certificate authority.

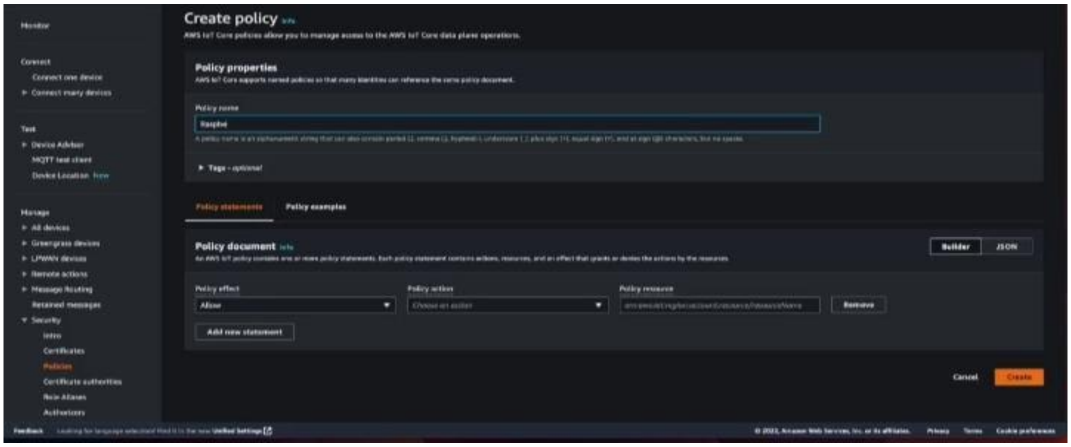
○ Use my certificate
Use a certificate signed by your own certificate authority.

○ Upload CSR
Register your CA and use your own certificates on one or many devices.

○ Skip creating a certificate at this time
You can create a certificate for this thing and attach a policy to the certificate at a later time.

Additional considerations:

In this post, we talked about ingesting data from diverse sources and storing it as S3 objects in the data lake and then using it to process ingested datasets until they're in a consumable state.



Create policy Info

AWS IoT Core policies allow you to manage access to the AWS IoT Core data plane operations.

Policy properties

AWS IoT Core supports named policies so that many identities can reference the same policy document.

Policy name: Raspbei

A policy name is an alphanumeric string that can also contain period (.), comma (,), hyphen (-), underscore (_), plus sign (+), equal sign (=), and at sign (@) characters, but no spaces.

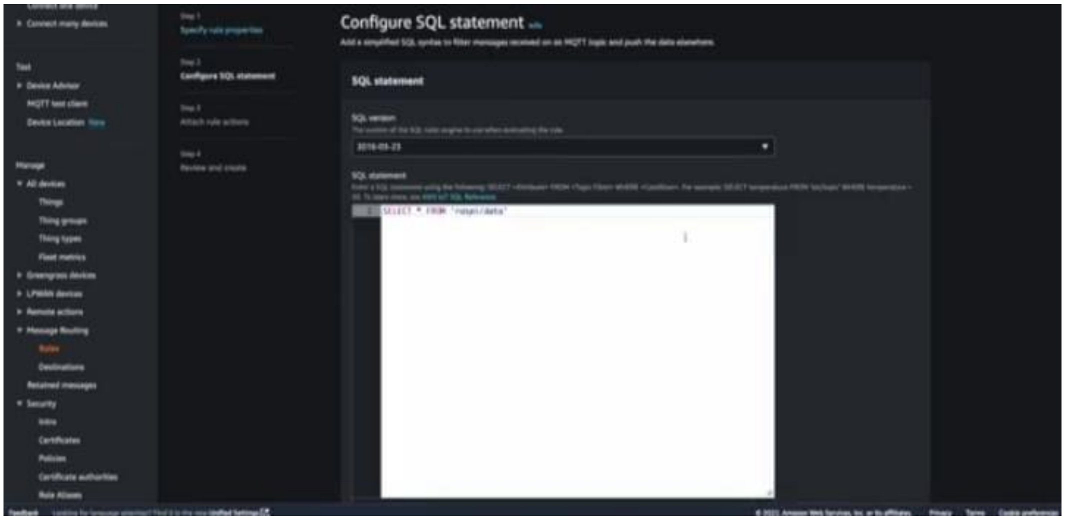
Policy document Info

An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

Policy effect: Allow

Policy action: Choose an action

Policy resource: arn:aws:iot:region:account:resource/resourceName



Step 1
Specify rule properties

Step 2
Configure SQL statement

Step 3
Attach rule actions

Step 4
Review and create

Configure SQL statement

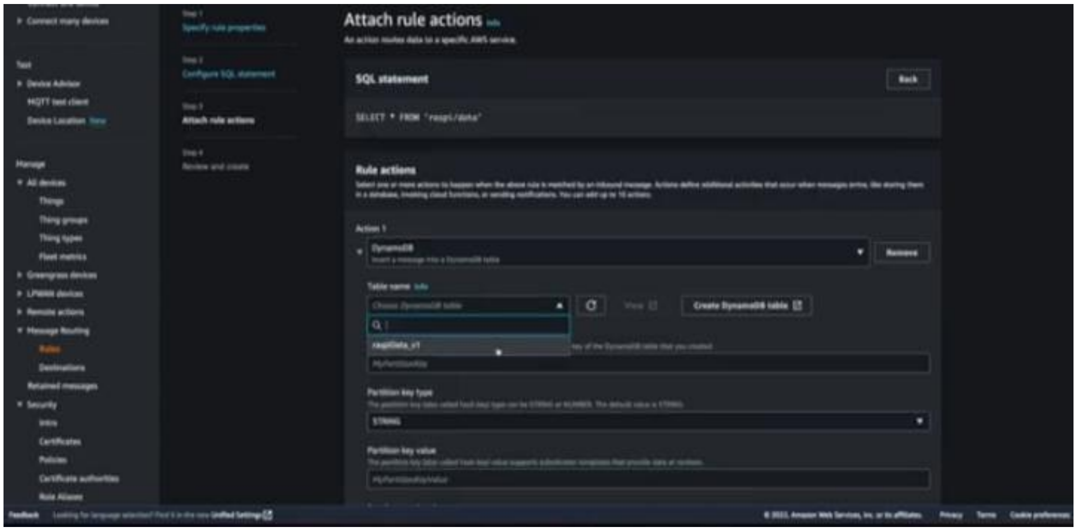
Add a simplified SQL syntax to filter messages received on an MQTT topic and push the data elsewhere.

SQL statement

SQL version: 2016-03-23

SQL statement

```
SELECT * FROM 'raspi/data'
```



Step 1
Specify rule properties

Step 2
Configure SQL statement

Step 3
Attach rule actions

Step 4
Review and create

Attach rule actions

An action routes data to a specific AWS service.

SQL statement

```
SELECT * FROM 'raspi/data'
```

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: DynamoDB
Insert a message into a DynamoDB table

Table name: Choose DynamoDB table
raspiData_v1

Partition key type: STRING

Partition key value

Conclusion:

serverless and managed services, you can build a modern, low-cost data lake centric analytics architecture in days. A decoupled, component-driven architecture allows you to start small and quickly add new purpose-built components to one of six architecture layers to address new requirements and data sources.