Disclaimer- Am Not expert in AWS Cloud Formation and AWS command Line. All this application is deployed suing AWS Console.

I have prepared the entire step by step development of the application in Youtube playlist. So please take a look at below playlist

<https://www.youtube.com/watch?v=Znq9-bTCpTk&list=PLrZZDAhaKWWJoFCFU4TVCAJ9D9lniJ8hK>

Below are the List of aws components used for this Application. We expect that person trying to re-produce below steps has basic knowledge on below topics.

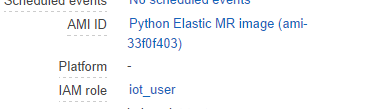
* AWS EC2
* AWS IOT
* AWS Kinesis Firehouse
* AWS Kinesis Analytics
* AWS Lambda
* AWS Kinesis Streams.
* AWS DynamoDB.

# Step1- Create EC2 Instance (t2.micro should Suffice)

EC2 instance is used to run the Mock Data generation program. We will execute the Python program which generates mocked iot data and publishes them to AWS IOT using AWS SDK.

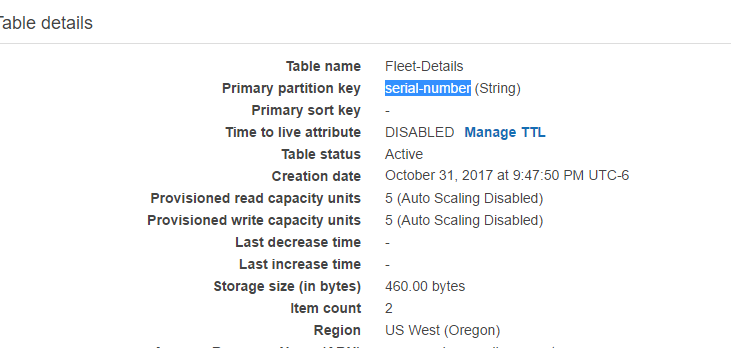
Our application is tested in us-west-2 Region using AMI “**ami-33f0f403**”. so please create Ec2 instance with mentioned AMI and ensure that IAM role is assigned to EC2 instance that has full permission on AWS IOT and DynamoDB. Preferable Give the Role with admin permission. Please ensure that the security group attached allows SSH connection from your computer.





# Step2 – Create A DynamoDB table

Create A DynamoDB which stores our application data. Please create with name “**Fleet-Details**” and primary partition Key as **“serial-number”** with String Datatype.



# Step 3: Load DynamoDB table and test Mock Data Generation Program.

Login into the ec2 instance created in the step1 and execute the below steps

* Install aws boto3 package in the instance. Below command for the same.

sudo pip install boto3

* Clone the git repository of application code.

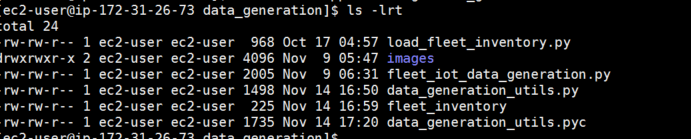
git clone <https://github.com/kalyanjanaki/aws-iot-app-challenge.git>

* Navigate to the directory of Data Generation. Below command for the same.

cd aws-iot-app-challenge/data\_generation/

* You should able to see below files in the directory.

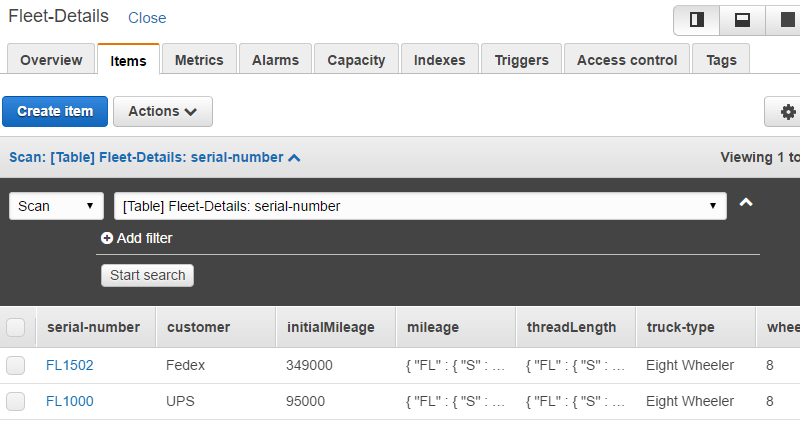
ls -lrt



* Execute Below program to load fleet details into Dynamo DB table.

python load\_fleet\_inventory.py

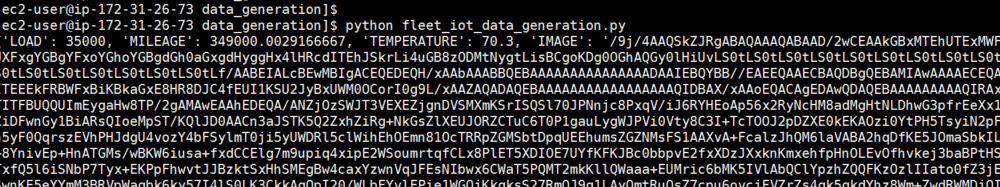
* Check if Dynamo-DB table is loaded with two rows.



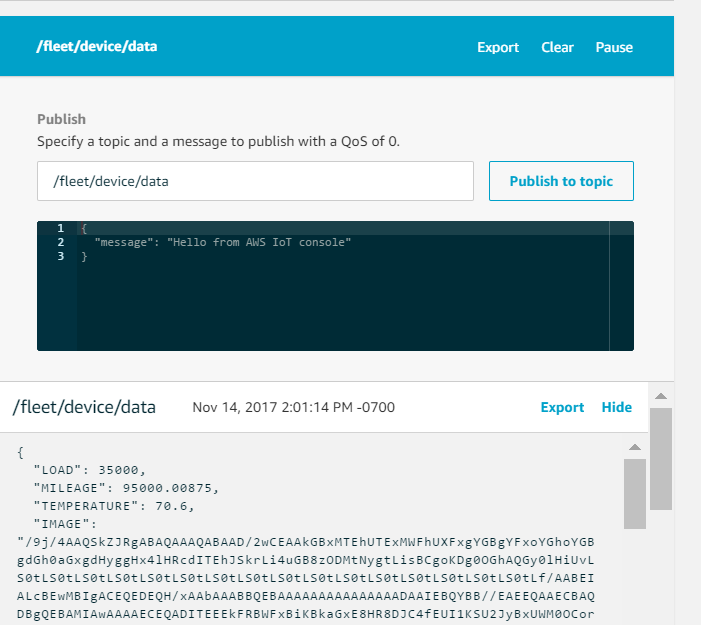
* Run the Data Generation program.

python fleet\_iot\_data\_generation.py

you should see continuous stream of data as below. Don’t worry about Huge Base64 Strings that is image Data as part of message.



* While Above program is running Please test if you are receiving messages in AWS IOT console. Login into AWS IOT Console and go the test section enter the topic as “/fleet/device/data” and subscribe You should be able to see messages as below.



* Go to Ec2 instance and close the data generation program using CTRL-C

# Step 4: Deploy Lambda Functions.

Ensure that you provide lambda functions below a admin role so that they have enough permissions.

**Function 1:**

Navigate to AWS lambda Console and create a lambda function and select the option author from scratch. Name the function as “Get\_Terrain\_from\_Image” . Select the runtime as “Python 2.7”. Use the code from below link.

<https://github.com/kalyanjanaki/aws-iot-app-challenge/blob/master/lambda/get_terrain_from_image/lambda_function.py>

NOTE:: This Function needs permission to call aws recognition API. So, ensure that lambda has sufficient IAM role.

**Function 2:**

Navigate to AWS lambda Console and create a lambda function and select the option author from scratch. Name the function as “Velocity-Visualize”. Select “Python 2.7” as runtime environment deploy it using the zip file provided in below link

<https://github.com/kalyanjanaki/aws-iot-app-challenge/blob/master/lambda/velocity_visualize/Velocity-visualize-2fce8865-2e5f-40a4-8038-5f14904f925e.zip>

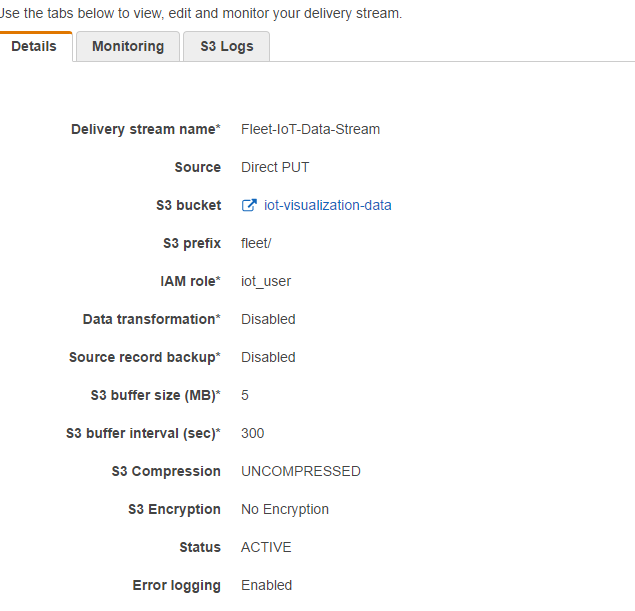
**Function 2:**

Navigate to AWS lambda Console and create a lambda function and select the option author from scratch. Name the function as “update\_fleet\_status”. Select “Python 2.7” as runtime environment deploy it using the zip file provided in below link

<https://github.com/kalyanjanaki/aws-iot-app-challenge/blob/master/lambda/update_fleet_status/update_fleet_status-e344a98e-b83f-4981-bda9-117aa56ee824.zip>

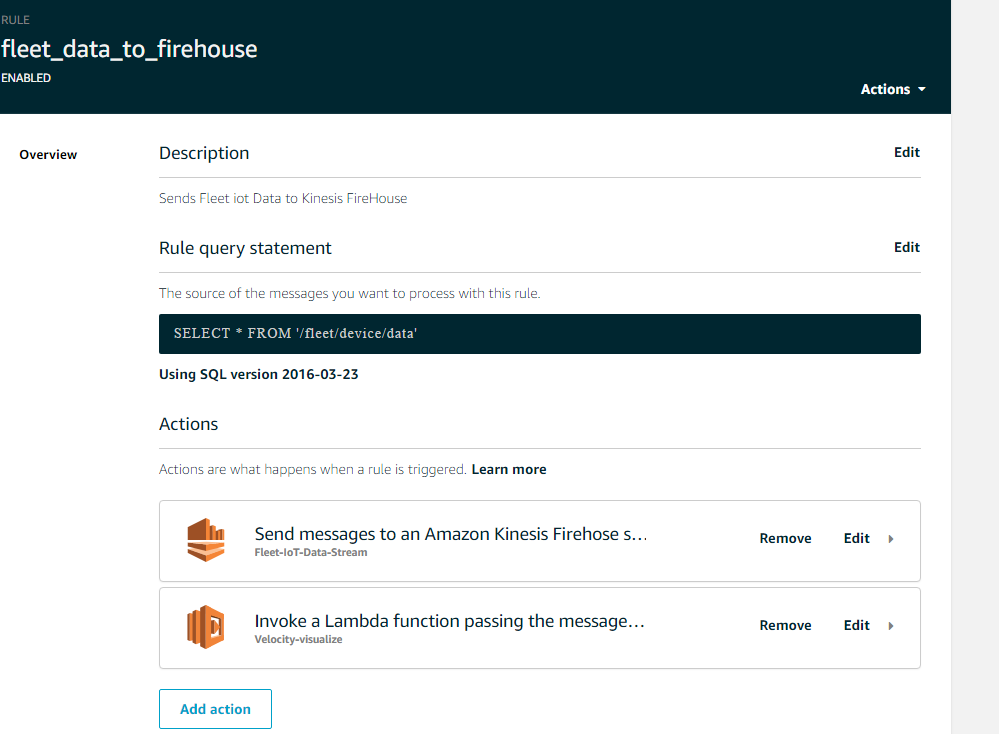
# Step 5: Create AWS Kinesis Firehouse Stream

Create a Kinesis Firehouse Stream with name “Fleet-IoT-Data-Stream” from AWS Kinesis Console. Select Destination as S3 and provide your own S3 bucket Details.



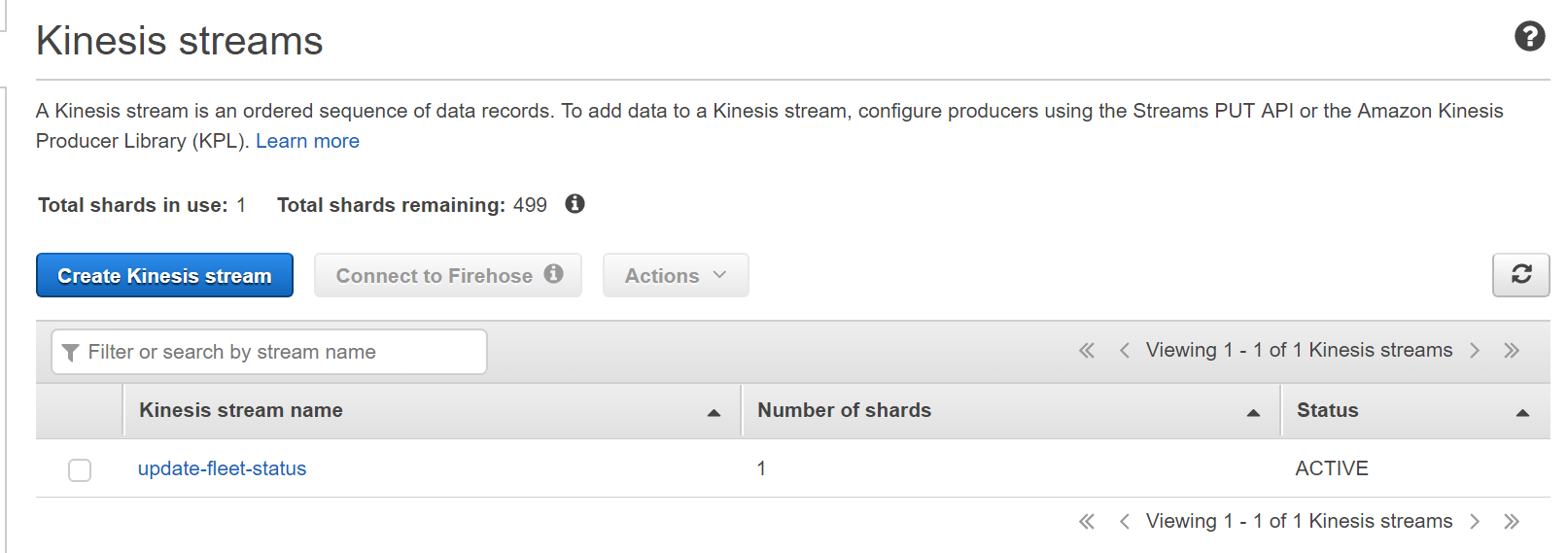
# Step 6: Deploy AWS IOT Rule Action

Navigate to AWS IOT Console and create a rule on the topic “/fleet/device/data”. Create the rule such that all the messages of that topic are send to a lambda function and Kinesis Firehouse. Below is the screen shot of the rule and action.

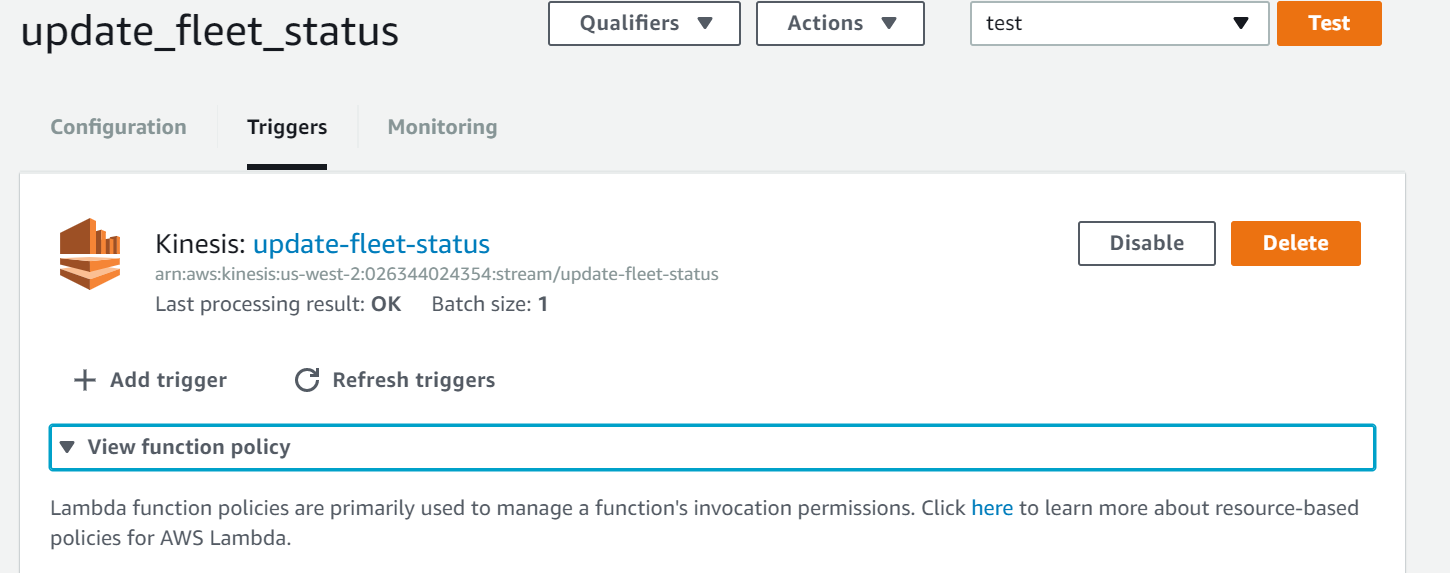


# Step 7: Create a Kinesis Stream and integrate with lambda Trigger.

Navigate to AWS Kinesis console and create kinesis-stream namely “[update-fleet-status](https://us-west-2.console.aws.amazon.com/kinesis/home?region=us-west-2#/streams/details?streamName=update-fleet-status)” as shown in below picture.



Once kinesis stream is successfully created integrate it with lambda function “update\_fleet\_status” created in step 4 . Create a lambda trigger for the function and assign to kinesis stream. Use offset as “LATEST” and batch size as 1. Below screen shot can help you.



# Step 8: Deploy Kinesis Analytics Application.

Navigate to AWS Kinesis Analytics Console and create new kinesis analytics application. Use source stream as Kinesis Firehouse stream created in step 5. Use lambda function 1 “Get\_Terrain\_from\_Image” created in the step 4 as preprocessor.

Use the sql code in the below link as kinesis sql code.

<https://github.com/kalyanjanaki/aws-iot-app-challenge/blob/master/kinesis-analysis/kinesis.sql>

Start the application.

# Step 9: Visualize the Dashboard and Real-Time changes in Dynamo-DB table

Run the Data generation program again in the ec2 instance as mentioned in the step 3.

Copy the html file in below link to your desktop and open the same in web browser. You should be able to see the live velocity visualization of the fleet and current tire wear status of the fleet as below.

Also see that mileage and Thread length fields of DynamoDB table are updated .