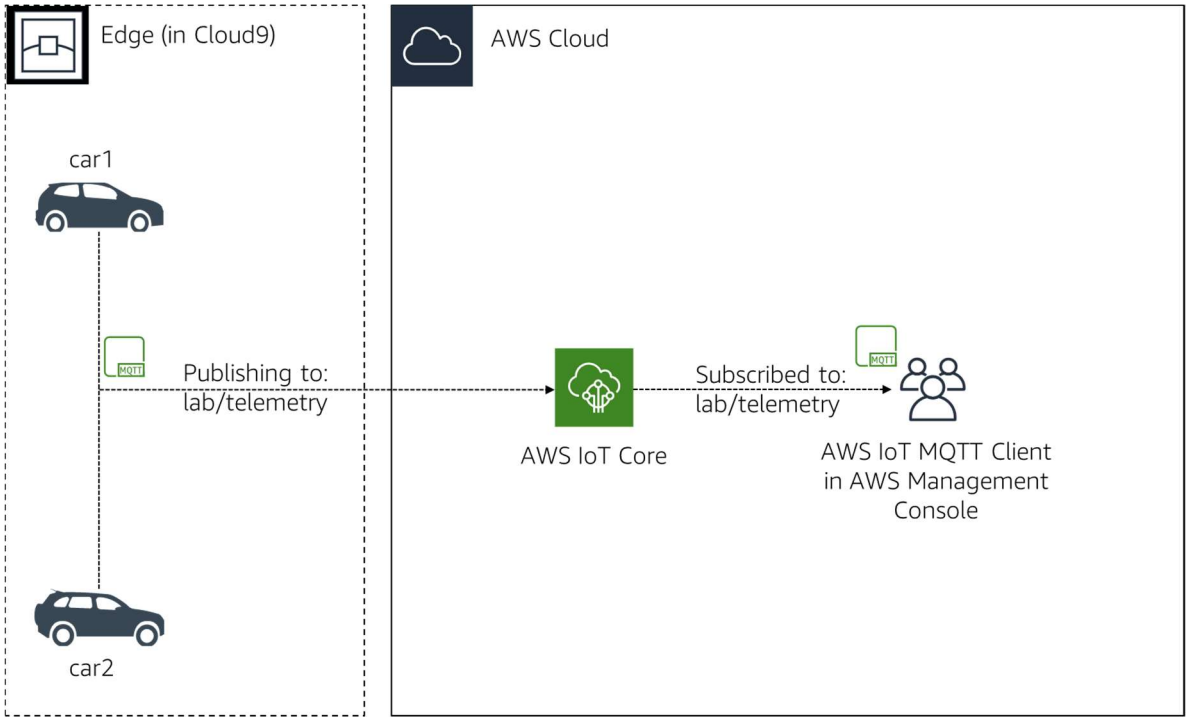
Exercise 6.1 - Connect 2 Car Things to IoT Core from Cloud9

In this exercise, you will create two Cars Things and connect them to AWS IoT Core service so they can send telemetry data on an IoT Topic that will be used in a later exercise. To connect those Cars, you will create an IoT Thing, Certificate and Policy. The Thing will represent a Car. The Certificate will be used to authenticate to AWS IoT Core and the Policy will define what your Car can do once authenticated. The first car will be created via the AWS Management Console while you will use the AWS Command Line Interface (CLI) to create the second Car.

The Cars will be simulated in a Cloud9 environment that you will need to create. You will download the code of the Cars, upload their IoT Certificates and start their engine. You will use the AWS IoT MQTT Client in the AWS Management Console to subscribe to the telemetry IoT Topic to confirm that the Cars are sending data.

The diagram below shows the resources and data flow that you will create in this exercise.



1. Create an IAM Policy

To begin, you will create an IAM User and Policy specific to this class so you can delete that user once you are done with the exercises.

In this section, you will create an IAM customer-managed policy. Customer-managed policies provide more precise control over your policies than policies managed by AWS. This policy will have permissions specific to the AWS resources you need for this course.

1. In the AWS Management Console, click **Services**, and then click **IAM** to open the IAM dashboard.
2. In the left navigation menu, click **Policies**.
3. Click **Create policy**.
4. Click the **JSON** tab.
5. In the editor text box, replace the sample policy with the following:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "greengrass:*",
        "iot:*",
        "iotanalytics:*",
        "cloud9:*",
        "lambda:*",
        "s3:*",
        "sns:*",
        "iam:*",
        "cognito-identity:*",
        "cognito-sync:*",
        "cognito-idp:*",
        "logs:*",
        "ec2:*",
        "cloudwatch:*",
        "kms:ListAliases",
        "kms:DescribeKey",
        "cloudformation:DescribeStackResources",
        "tag:getResources"
      ],
      "Resource": "*"
    }
  ]
}
```

6. Click **Review Policy**.
7. For **Name**, enter `labIoTPolicy`.
8. Click **Create policy**.

   You successfully created an IAM policy.

2. Create an IAM user and attach a policy to the user.

In this section, you will create an IAM user and attach a policy to the user.

1. In the AWS Management Console, click **Services**, and then click **IAM** to go to the IAM dashboard.
2. In the left navigation menu, click **Users**.
3. Click **Add user**.
4. In the **User name** text box, enter `labIoTUser`.
5. For **Access type**, select **AWS Management Console access**.
6. For **Console password**, choose **Custom password** and enter a password of your choosing. **Note the password**.
7. Remove the **check mark** next to **User must create a new password at next sign-in**.
8. Click **Next: Permissions**.
9. In the **Set permissions** section, click **Attach existing policies directly**.
10. In the search text box for **Filter**, enter `labIoTPolicy`.
11. Put a check mark next to **labIoTPolicy** in the filtered list.
12. Click **Next: Tags**.
13. Click **Next: Review**.
14. Review the information, and click **Create user**. You should see a success message.
15. Note the **sign-in URL** in the success message at the top. This is a special URL for IAM users, which includes your account ID.

16. Click on the **sign-in URL** in the success message at the top. This will log you out.
17. **Sign in** as the `labIoTUser` IAM user.

3. Create an AWS Cloud9 environment

In this section, you will create an AWS Cloud9 environment.

1. In the AWS Management Console, click **Services**, and then click **Cloud9** to open the Cloud9 dashboard.
2. Make sure you are in the **Frankfurt, Ireland, N. Virginia, Ohio, Oregon or Tokyo** Region. Those are, at the time of writing the exercises, the only regions with all the services that will be used: Amazon Cognito, Amazon EC2, Amazon S3, Amazon SNS, AWS Cloud9, AWS IAM, AWS IoT Analytics, AWS IoT Core, AWS IoT Greengrass, AWS Lambda. Since all resources must be in the same region for the exercises to work, you must use a region where all those services are available. You can find a list of the services available per region at this link.[1]
3. Click **Create environment** at the top-right corner.
4. For **Name**, enter `IoTOnAWS`.
5. Click **Next step**.
6. On the Configure Settings page, leave the default settings, and click **Next step**.
7. Review the details and click **Create environment**. This should launch your AWS Cloud9 environment within a few minutes.

   Note that this Cloud9 instance will automatically shutdown after 30 minutes if it's not used. All of your work will be saved and brought back to what it was if you were to re-open it.

4. Setup your Cloud9 Environment, download the Car code and the AWS IoT CA Public Cert

In this section, you will install the Node package for the Car code to work, download the Car code and setup your repository structure for the cars.

[1] https://aws.amazon.com/es/about-aws/global-infrastructure/regional-product-services/

1. Install the AWS IoT Device SDK Node package by running the following command in your AWS Cloud9 **terminal**. You can find that terminal at the bottom of the page. There is a *bash* tab with the prompt *labIoTUser:~/environment: $*. You can adjust the size of that screen like you would in a normal IDE: put your cursor above the tab and select&drag to increase/decrease the space.

```
npm install aws-iot-device-sdk
```

   You can ignore the warnings that there are no package.json files.
2. Create the repository structure for the Car application. As there will be 2 Cars, you will create 2 folders. Run the following commands in your AWS Cloud9 **terminal**.

```
mkdir ~/environment/car1; mkdir ~/environment/car2
```

3. Download and copy the application code in each Car folder by running the following commands in your AWS Cloud9 **terminal**:

```
cd ~/environment

wget https://aws-tc-largeobjects.s3.amazonaws.com/OTP-
AWS_D5-2019/v1.0/code/exercise-1.1.js

cp exercise-1.1.js car1/

cp exercise-1.1.js car2/

rm exercise-1.1.js
```

4. Download the AWS IoT Certificate Authority Public Certificate that will be used in the code later and that will sign the IoT Certificates you will create in the next section. To do so, execute the following commands in your Cloud9 **terminal**:

```
cd ~/environment
wget -O root-CA.crt
https://www.amazontrust.com/repository/AmazonRootCA1.pem
```

5. Create Car 1 IoT Thing, Certificate and Policy

In this section you will use the AWS Management Console to create all of the resources required for your Car 1 to connect to AWS IoT. This includes the the Car Thing, the Certificate and the Policy. We could use the wizard to create many of these resources, but instead, you will do it manually to see how each of the components is attached to each other. You will finish by uploading the Certificate and Private Key that you generated as part of these steps to Cloud9.

5.1 Create an IoT Thing

In this section, you will create a Thing representing your Car.

1. You may have lost your AWS Management Console and only see the Cloud9 console at this moment. To get back to the AWS Management Console without having to retype the URL, click on the cloud icon with the number 9 in it at the top left corner of the page and select **Go To Your Dashboard**.
2. In the AWS Management Console, click **Services**, and then click **IoT Core** to open the the IoT Console.
3. Make sure you are in the same **Region** as your Cloud9 instance.
4. Click **Get started**.
5. Expand **Manage** in the left menu.
6. Click **Things**.
7. Click **Register a thing**.
8. Click **Create a single thing** (any of the two buttons with that name works).
9. For **Name**, enter `car1` and click **Next**.
10. Click **Create thing without certificate** so that you skip the creation of the Certificate via the wizard.

    The car1 Thing has now been created.

5.2 Create an IoT Policy

In this section, you will create a Policy for authorization purposes that will be used in the next section.

1. Expand **Secure** in the left menu.
2. Click **Policies**.
3. Click **Create a policy**.
4. For **Name**, enter `labPolicy`.
5. Click **Advanced mode**.
6. **Replace** the sample policy with the following policy which authorize to Connect to your AWS IoT Core endpoint, to Publish and Subscribe to an IoT Topic, Receive messages from AWS IoT once subscribed and use the Discover API from Greengrass which will be used in a later exercise.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect",
                "iot:Publish",
                "iot:Subscribe",
                "iot:Receive",
                "greengrass:Discover"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

7. Click **Create**.

   You now have a Policy that provides authorizations.

5.3 Create an IoT Certificate

In this section, you will create a Certificate that will be used for Authentication.

1. Expand **Certificates**.

2. Click **Create a certificate**.
3. Click **Create certificate** to automatically generate a Certificate, a Public Key and a Private Key using AWS IoT's Certificate Authority that you will then download. Note that you could create your own Certificate Authority. You could also create your own Private Key on your end and generate a Certificate Signing Request that you would upload here to be signed by AWS IoT's Certificate Authority. Both could work with the exercise, but you would have to adapt some of the later commands. For simplicity, use the One-click certificate creation.
4. Make sure that you do the following steps before going to the next screen as you would have to restart the creation of this Certificate.
    1. Click the **Activate** button to activate the Certificate so it can be used later to connect to AWS IoT Core by your Thing.
    2. Click the **Download** link next to **A certificate for this thing**.
    3. **Rename** this file to `certificate.pem.crt`.
    4. Click the **Download** link next to **A private key**. Note that you won't need the public key.
    5. **Rename** this file to `private.pem.key`.
5. Click **Done**.

    You now have a Certificate and Private Key that can be used to connect to your AWS IoT Core endpoint. However, this is only for authentication, you don't have any authorization yet associated to this Certificate. You may see that the Certificate is inactive, if you refresh the page, it should show as active. This will be done next.

5.4 Associate the Policy and Thing to your Certificate

In this section, you will attach a Policy to your Certificate to add authorizations and attach the car1 Thing that is related to this Certificate.

1. Click on the certificate you just created.
2. Click **Actions > Attach policy**.

3. Put a **check mark** next to **labPolicy** and click **Attach**.
4. Click **Actions > Attach thing**.
5. Put a **check mark** next to **car1** and click **Attach**.
6. If you click on Policies or Things on the left menu, you can see that both the labPolicy Policy and car1 Thing have been attached.

5.5 Upload your Certificate and Private Key to Cloud9

In this section, you will upload the Certificate and Private Key that are now associated to car1 into Cloud9 using the upload feature.

1. You may have closed your Cloud9 tab. If you did so, in the AWS Management Console, click **Services**, and then click **Cloud9** to open the Cloud9 dashboard. Click **Open IDE** under the IoTOnAWS environment card.
2. In the left menu, click the **car1** folder to select it.
3. Click **File > Upload Local Files...**.
4. Make sure that Upload to folder is: */car1*.
5. Click **Select files**. Note that the button may appear in grey, it isn't disabled and you can click on it.
6. **Browse** to the **Certificate** you have downloaded and renamed earlier: certificate.pem.crt. **Select it** and click **Open**. The Certificate should now be uploaded to your car1 folder.
7. Click **Select files** again.
8. Browse to the Private Key you have downloaded and renamed earlier: private.pem.key. **Select it** and click **Open**. The Private Key should now be uploaded to your car1 folder.
9. Click the **x** icon next to **Upload Files** to close that window.

   Both car1 Certificate and Private Key should now be in the *car1* folder.

6. Create Car 2 IoT Thing, Certificate and Policy

In this section you will use AWS CLI from the Cloud9 terminal to create all of the resources required for your Car 2 to connect to AWS IoT. This includes the Car Thing and the Certificate. The Policy has already been created from the previous step, so you will re-use it.

6.1 Creation of car2 IoT Core resources using the CLI

1. In your Cloud9 **terminal**, enter the following commands to create the *car2* Thing:

```
cd ~/environment/car2
aws iot create-thing --thing-name car2
```

2. To create the Certificate, enter the following command:

```
aws iot create-keys-and-certificate --set-as-active --certificate-pem-
outfile certificate.pem.crt --private-key-outfile private.pem.key
```

This command will place the Certificate and Private Key in the certificate.pem.crt file and private.pem.key respectively. It will also output the certificateArn which you will re-use in the next command.

3. To attach the Policy to the Certificate, enter the following command. **Replace certificateArn_changeme** with the value of the attribute certificateArn from the output of the previous command. It should be similar to: *arn:aws:iot:us-east-1:1234567890:cert/0f11db22dafacda87be0940dd5b2e010635916f541461ccf2d1c56ced0f343ee*

```
aws iot attach-policy --policy-name labPolicy --target
certificateArn_changeme
```

The command should not return anything if it worked.

4. To attach the car2 Thing to the Certificate, enter the following command. **Replace certificateArn_changeme** with the value of the attribute certificateArn from the output of the certificate creation command (2 steps above). It should be similar to: *arn:aws:iot:region:1234567890:cert/0f11db22dafacda87be0940dd5b2e010635916f541461ccf2d1c56ced0f343ee*

```
aws iot attach-thing-principal --thing-name car2 --principal
certificateArn_changeme
```

The command should not return anything if it worked.

6.2 Validation of creation of car2 IoT resources

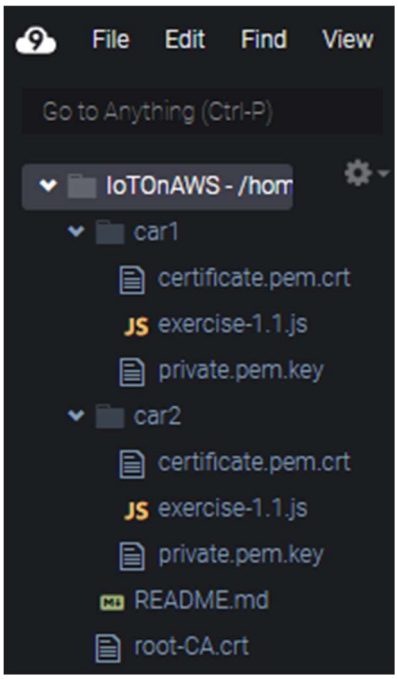
In this section, you will validate that all the resources for the creation of car2 have been completed successfully. If you are missing any resources, you may want to try the commands above again, but first, **Delete** the car2 Thing using the Console by selecting it and clicking **Actions > Delete** and try the steps from Section 6.1 again. If you still can't get those to work, use Section 5 to create everything from the AWS Management Console.

1. In the AWS Management Console, click **Services**, and then click **IoT Core** to open the IoT Core console.
2. Expand **Manage** and click on the **Things** menu.
3. You should see *car2* listed. If not, refresh the page as your browser may have cached the previous version of this page.
4. Click **car2**.
5. Click **Security**.
6. You should see a Certificate card.
7. Click on that Certificate card.
8. Click on **Policies**.
9. You should see the labPolicy attached.

You have now created the car2 Thing, its Certificate and Private Key to authenticate and attached the labPolicy to authorize the commands to execute later.

7. Execute the code and validate telemetry

At this point, you should have a directory structure that looks like the following in your Cloud9 environment:



In this section, you will execute the code of both cars and validate that telemetry data is sent by both cars using the AWS IoT MQTT Client in the AWS Management Console.

The code (exercise-1.1.js) requires one more resource to communicate with AWS IoT and that is your specific AWS IoT Endpoint. It will be stored in a file that will be used in all of the other exercises. To authenticate to your AWS IoT Endpoint, it will use the *certificate.pem.crt* (Certificate), *private.pem.key* (Private Key) and *root-CA.crt* (Certificate Authority public certificate). It then connects to AWS IoT and starts publishing random telemetry data every 5 seconds to the *lab/telemetry* IoT Topic.

Feel free to read the code to understand what is happening. Comments have been added to the code so it can be self explanatory.

7.1 Execute the code

In this section, you will first obtain your specific AWS IoT Endpoint that was automatically created for you when you created your first IoT Thing. This normally takes a few minutes to complete. Depending how fast you went with the steps above, you may have to wait a few minutes if the next command doesn't return anything. You will then open two Cloud9 terminals to simulate a connection to AWS IoT from both cars. From each terminal, you will run the code which will use its appropriate Certificate and Private Key as you placed them in their respective folders.

1. In the Cloud9 **terminal**, enter the following command to get your specific AWS IoT Endpoint which will then be saved in the endpoint.json file:

   ```
   aws iot describe-endpoint --endpoint-type iot:Data-ATS >
   ~/environment/endpoint.json
   ```

   The command above will not output anything if it worked. Feel free to look at that file as it will be visible on the left side of the editor.

2. In the Cloud9 environment, click the **circled +** icon that is next to your current terminal and select **New Terminal**. You now have 2 different terminals.

3. In the **left terminal**, execute the following commands to start the code for car1. Car1's Certificate and Private Key, the Root Certificate Authority and your specific AWS IoT Endpoint will all be read from the files you created.

   ```
   cd ~/environment/car1
   node exercise-1.1.js
   ```

   You should see the following:

   ```
   Connected to AWS IoT
   Sending car telemetry data to AWS IoT for car1
   Sending car telemetry data to AWS IoT for car1
   ...
   ```

4. In the **right terminal**, execute the following commands to start the code for car2. Car2's Certificate and Private Key, the Root Certificate Authority and your specific AWS IoT Endpoint will all be read from the files you created.

   ```
   cd ~/environment/car2
   node exercise-1.1.js
   ```

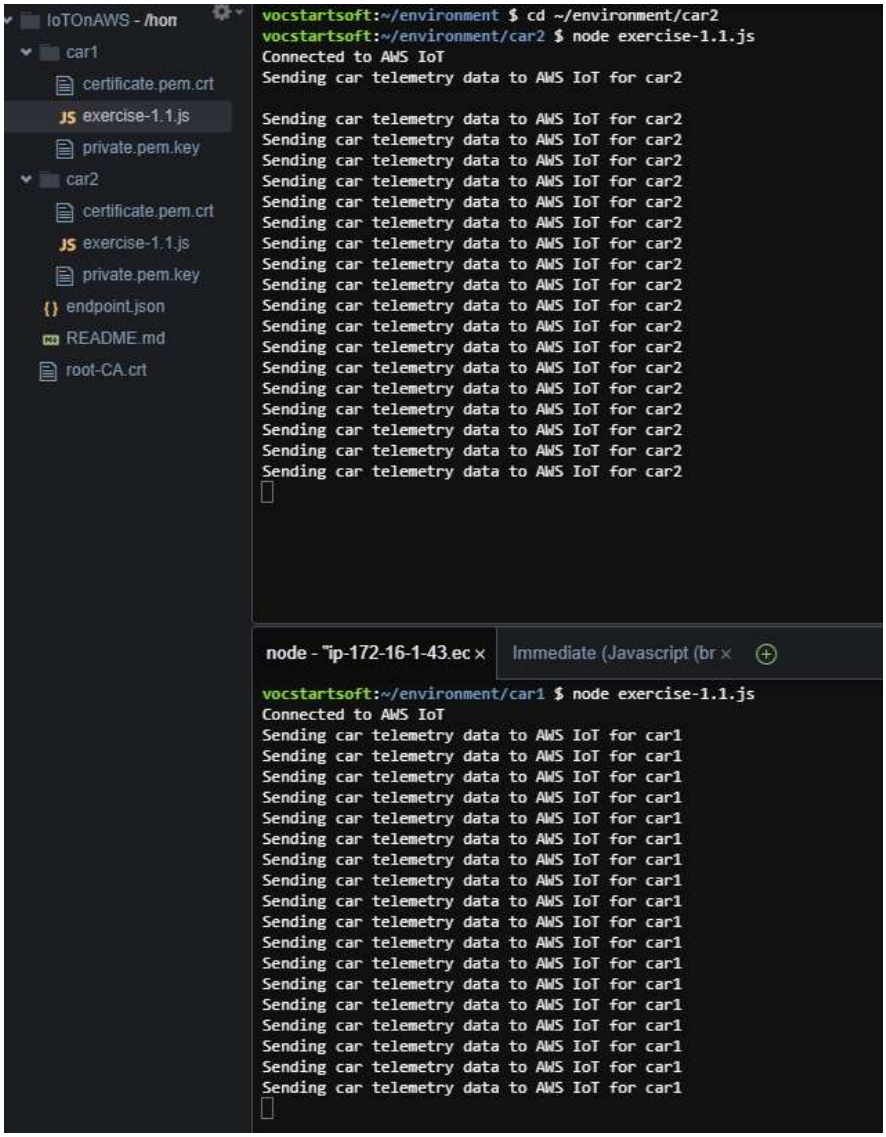
   You should see the following:

```
Connected to AWS IoT

Sending car telemetry data to AWS IoT for car2

Sending car telemetry data to AWS IoT for car2

...
```



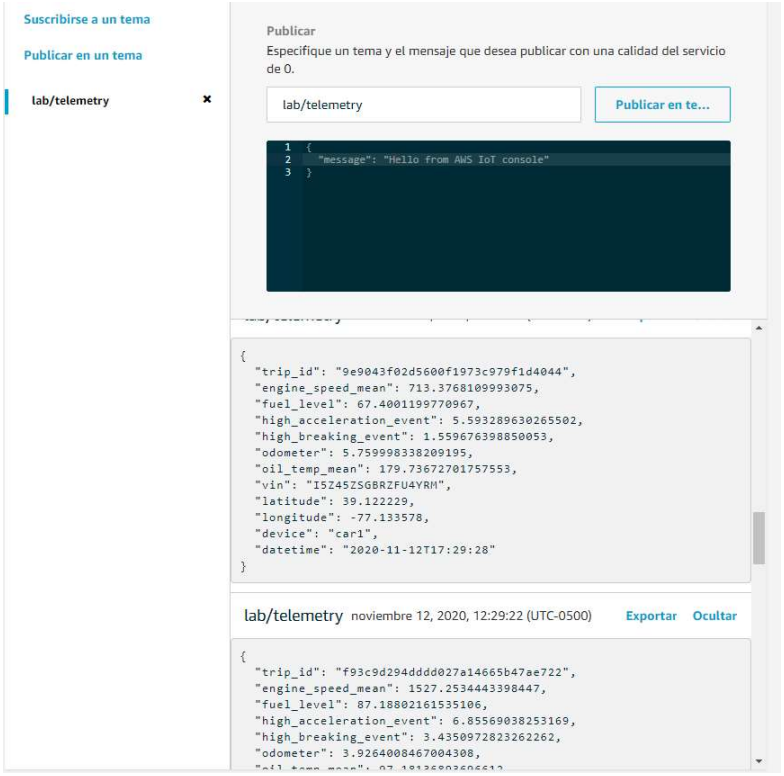
Both cars are now successfully sending telemetry data to the *lab/telemetry* IoT Topic.

7.2 Subscribe to the lab/telemetry Topic

In this section, you will use the AWS IoT MQTT Client in the AWS Management Console to subscribe to the *lab/telemetry* IoT Topic. While connected to this MQTT Client, it does consume connection minutes. The free tier is generous and provides 2,250,000 minutes of connection for free, but you should make sure to disconnect when you are not using it by browsing away from that page.

1. In the AWS Management Console, click **Services**, and then click **IoT Core** to open the IoT Core console.
2. Click **Test** in the left menu. It will open an AWS IoT MQTT Client where you can interact with any Topic that you have access to. This Client will automatically connect to your IoT Endpoint.
3. In the **Subscription topic**, enter `lab/telemetry`.
4. Click **Subscribe to topic**.

   In the next 5 seconds, you should start seeing data being published by both cars in the interface. You can see which car is sending the data by looking at the *device* attribute.



8. Delete the resources created in this exercise

While there are no connections nor data being transmitted to the IoT service, there will not be any charge for this exercise for that service.

The Cloud9 environment uses a t2.micro EC2 instance and an 8GiB Elastic Block Storage (EBS) volume which is what you are being charged on for the usage of Cloud9. The t2.micro has 750 hours of free utilization per month under the free tier for the first 12 months after opening your account. You have 30GiB of space for EBS covered under the free tier for the first 12 months of the opening of your account. If you are no longer under the free tier, you will incur a charge while the EC2 instance is running for the t2.micro instance and a charge for the EBS volume as long as your environment exists.

All of the other exercises will use the resources created as part of this exercise. The recommendation is to keep the IoT resources for the future exercises and to let Cloud9 stop by itself after 30 minutes of inactivity so you can keep the environment. If you decide not to keep some of those resources, you will have to do this exercise again before the other exercises.

If you would prefer to remove all of your resources. Follow the steps hidden below:

Expand for instructions on how to delete all your resources (not recommended).

8.1 Stop the cars

1. **Press Ctrl-c** in each of the Cloud9 **terminal** to stop them from interacting with AWS IoT.

8.2 Stop the MQTT Client

1. **Navigate away** from the **AWS IoT MQTT Client** page to disconnect from the client.

8.3 Stop the Cloud9 environment

The Cloud9 environment will automatically shut down after 30 minutes of inactivity. For your Cloud9 environment to be considered inactive, you need to close the browser tab. All of the settings will be saved.

1. Close the **browser tab** where your environment was running.

As the operating system is Amazon Linux, you are billed by the second during those 30 minutes of inactivity. If you are under the free tier, this would be covered. If you are no longer under the free tier, you can force a stop of the EC2 instance that runs your Cloud9 environment. This will have no effect on the future exercises.

1. In the AWS Management Console, click **Services**, and then click **EC2** to open the EC2 console.
2. Click **Instances** in the left menu.
3. Select the EC2 Instance that has a name that starts with **aws-cloud9-IoTOnAWS**.
4. Click **Actions > Instance State > Stop instance**