Workshop Streaming Data with Amazon Kinesis

SIG Event Architecture 15 mei 2018

Inhoud

Prerequisites

For following this workshop you need the following:

- An Amazon AWS account with access to Kinesis, EC2, S3
  - Use your Amis AWS account if you have any
- AWS SDK for Python for running the sample code
- AWS command line interface CLI (see for more information: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html)
- For running Python you need Python 3.6.4 or up. Python can be found at https://www.python.org/
  NOTE: When installing Python: Select Add Python to the Path option!
- An IDE for Python
  - for example: Pycharm, download at https://www.jetbrains.com/pycharm/download/ or use the version from the share

Files

Any support files can be found in the Amis Github repository: https://github.com/AMIS-Services/sig-streaming-data-may-2018.

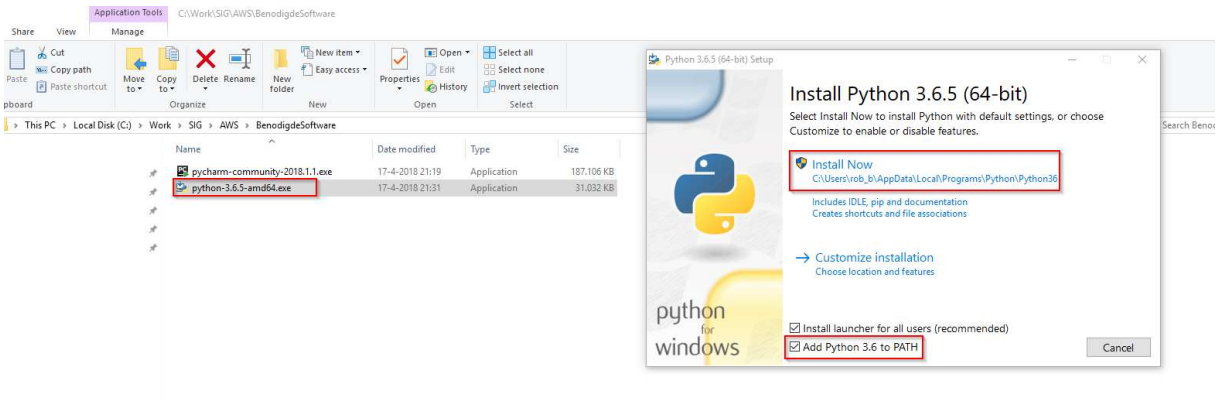
Reference

- Kinesis Developer Guide:
  https://docs.aws.amazon.com/streams/latest/dev/kinesis-dg.pdf
- Kinesis Analytics Developer Guide:
  https://docs.aws.amazon.com/kinesisanalytics/latest/dev/analytics-api.pdf

Software installation

Python 3.6

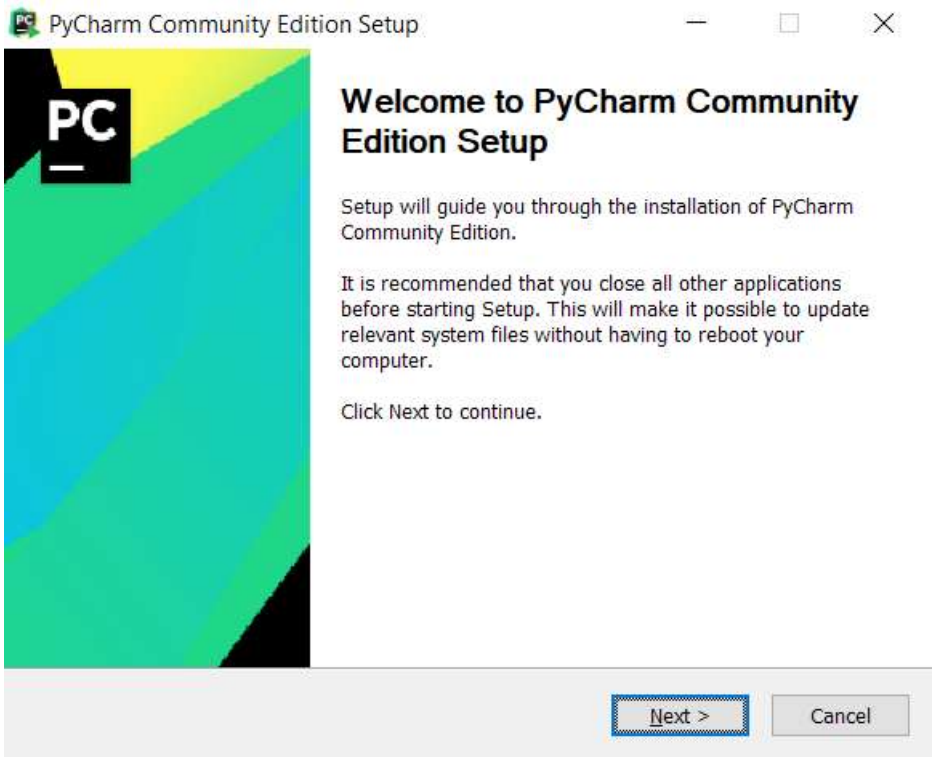
1. Download *python-3.6.5-amd64.exe*

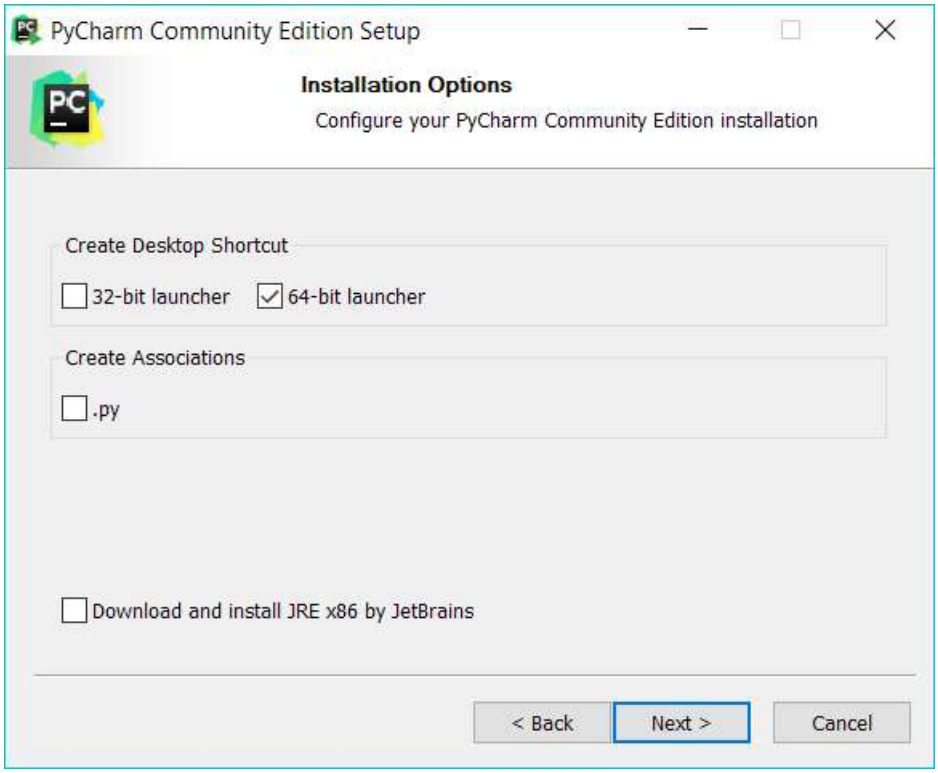


2. Run the installer *python-3.6.5-amd64.exe*. select the **Add Python 3.6 to PATH** checkbox!
3. Choose *Install Now*

Pycharm Community Edition

1. Run the install *pycharm-community-2018.1.1.exe* and choose the 64-bit launcher





Optionally check .py to start PyCharm when double clicking op *.py files.

Just press Next, select the directory and install.

Install AWS CLI

Using pip

The primary distribution of the AWS CLI is by using pip (the Python package installer). If you you want to use this you need Python 2.6.5 or up or Python 3.3 or up. Just take the latest 3.6.x version.
If you have Python and pip installed, open a command shell (cmd or powershell on Windows) and type:

```
pip install awscli --upgrade --user
```

The `--upgrade` option tells `pip` to upgrade any requirements that are already installed. The `--user` option tells `pip` to install the program to a subdirectory of your user directory to avoid modifying libraries used by your operating system.

Post install

After installation you need to add de AWS CLI to your command line path. Follow the instructions here if you do not know how:

- Linux – Adding the AWS CLI Executable to your Command Line Path
- Windows – Adding the AWS CLI Executable to your Command Line Path
- macOS – Adding the AWS CLI Executable to your Command Line Path

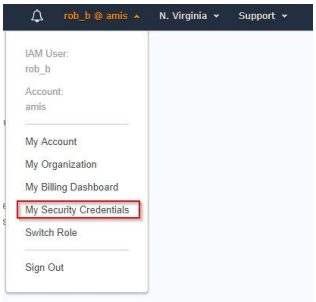
Type: `aws -version` to check if the installation was successful.
Type: `pip install awscli --upgrade --user` to upgrade an existing installation.
Type: `pip uninstall awscli` to uninstall the AWS CLI.

Configuration

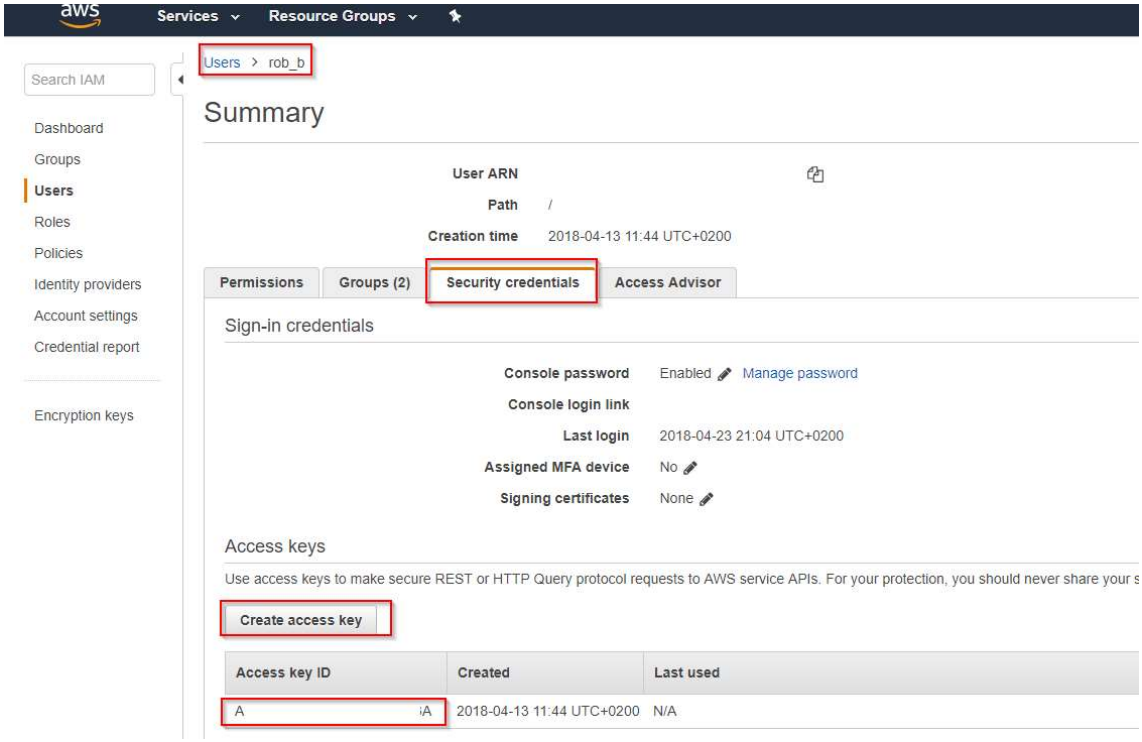
For the AWS Credentials, you must login to the Amazon Console and go to IAM (if you have access to it).
We will provide the access key and secret access key. Most users will not have access to the IAM console to do this.

If you do have access: these are the steps to generate the access key:

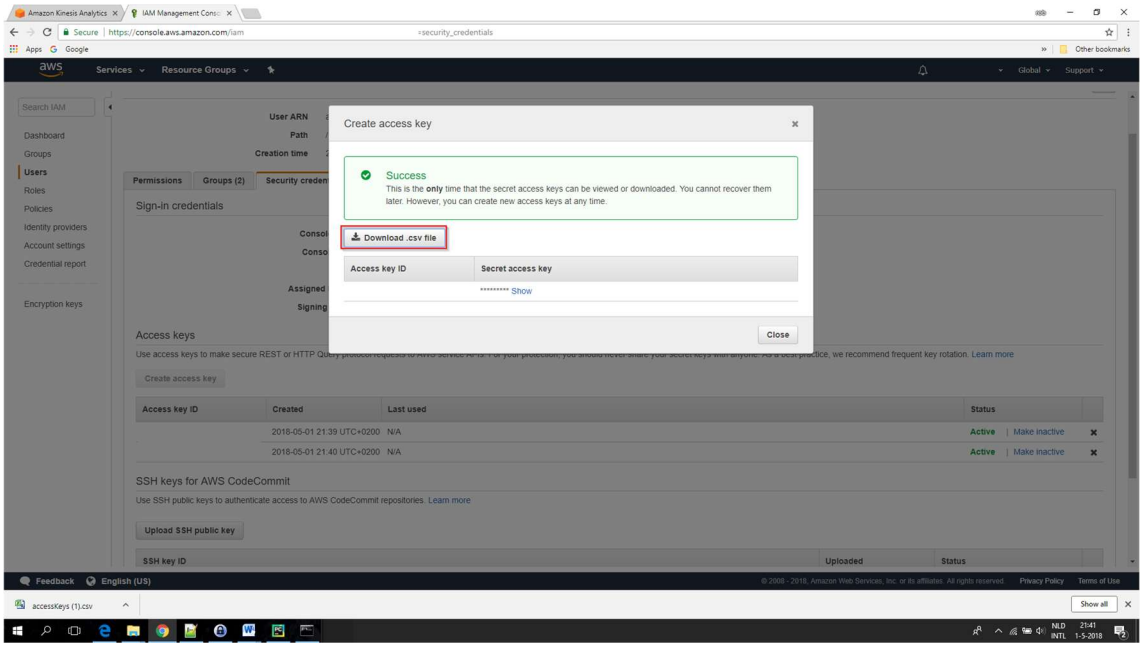
Select in the topbar your name → Select My Security Credentials



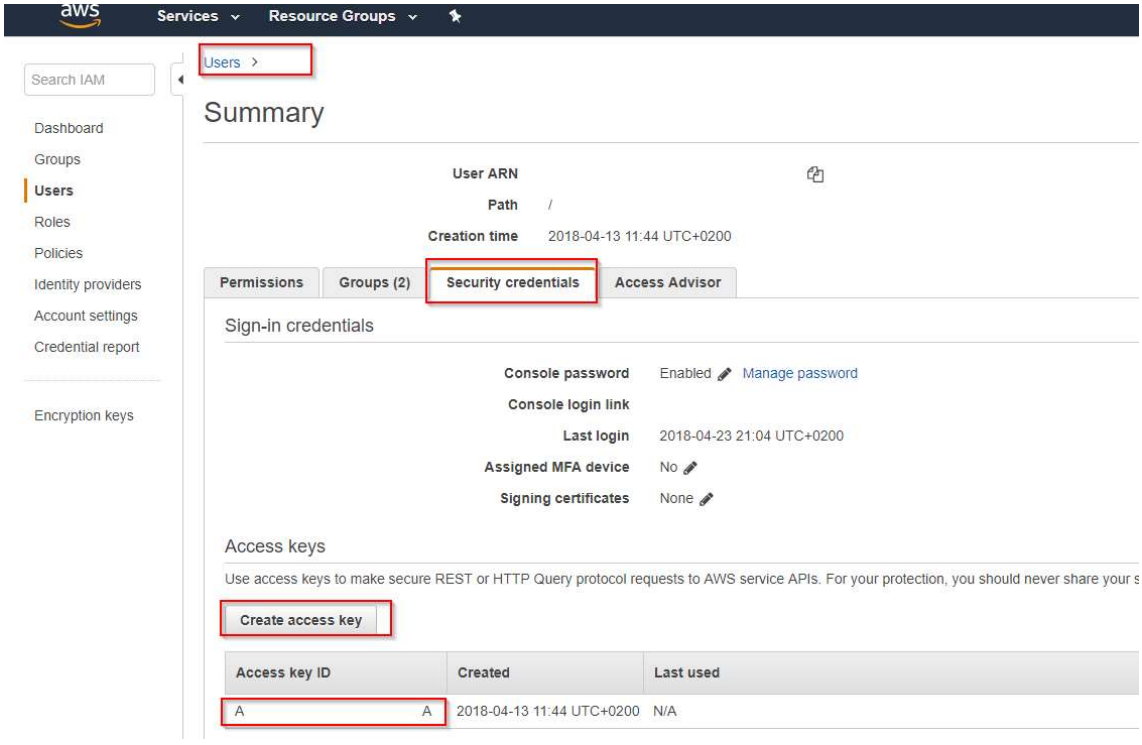
Select Users from the left menu → select your user name from the list →
The screen below will be shown



You need to generate the access key again because the *secret access key* will only be shown during the creation of the key. Don't forget to save the keys in the csv-file, see screenshot below.

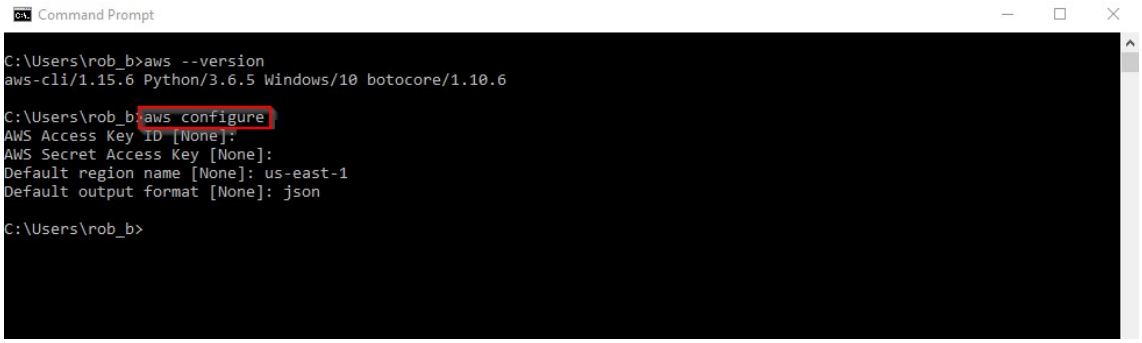


Or start the IAM service and open the Users page and locate your user id.



Select the Security credentials tab and press Create accessKey.
Now you will have the option to show the Key and Secret. Save them and use it when running aws configure. After you leave the page, you won't be able to see the Secret anymore. Only the Access key ID will be shown.
Type: `aws configure` to start the configuration

You need to enter the AWS Access Key ID, AWS Secret Access Key, the default region name (`us-east-1`) and default output format (`json`).

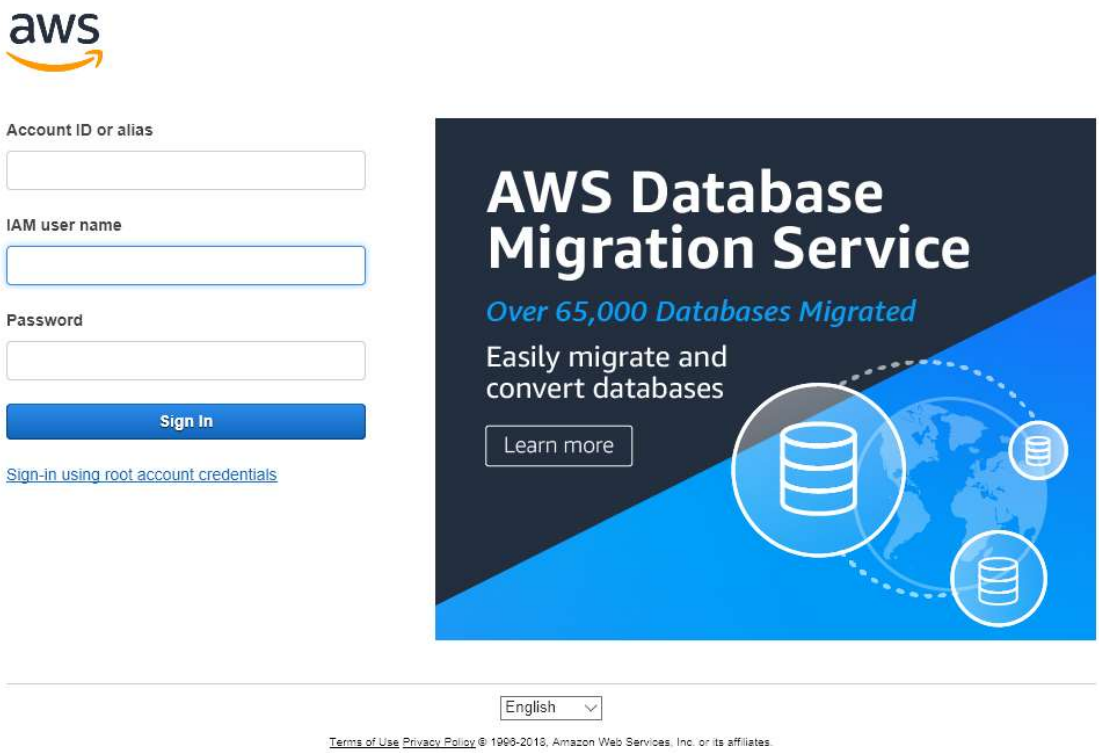


Now your Security credentials are stored (in a local file in the subfolder .aws in your profile/user directory) so `awscli` automatically has access to this key.

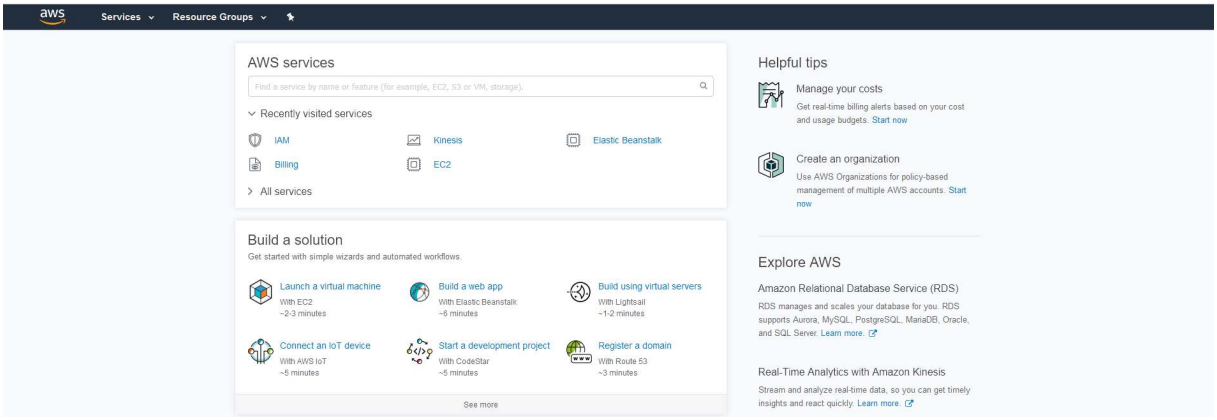
AWS console

Log in op de AWS console: https://console.aws.amazon.com/console/home

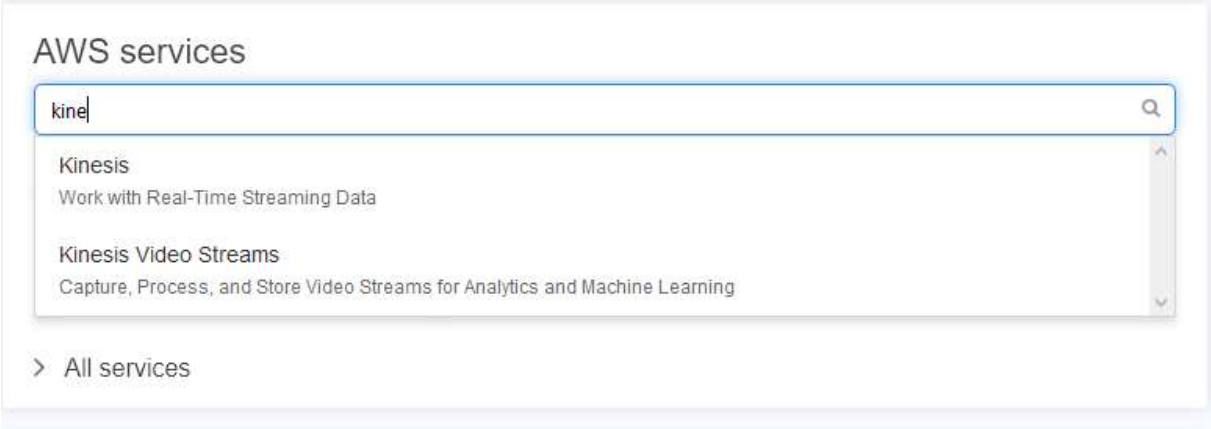
Enter your AWS Account ID, IAM username and password.



You will arrive at the home page of the console.



Under recently visited services, you will find the services you last visited.
If you click on > All Services, you will see all other services. Note that you see every possible service, that does not mean you have access to it!
You can search services by start typing in the search box.

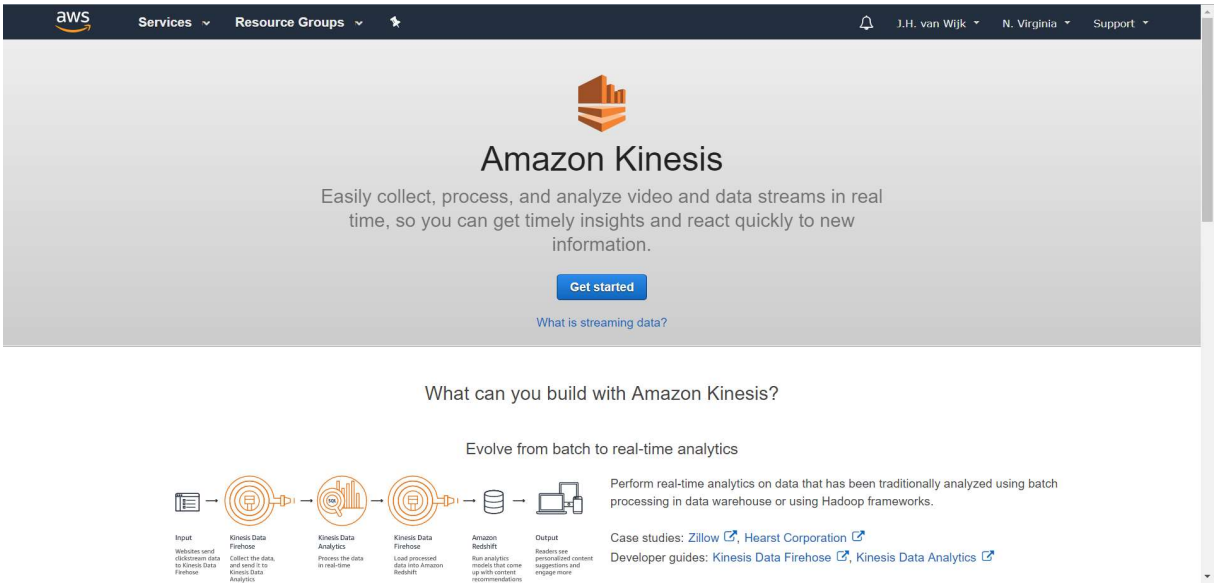


Start typing kinesis.

Select Kinesis.

The Kinesis homepage is now shown. You also navigate to it directly by using the url: https://console.aws.amazon.com/kinesis

If there is no Kinesis service created yet, you will see an informational page.



Select the correct region: us-east-1 (if this not already set):

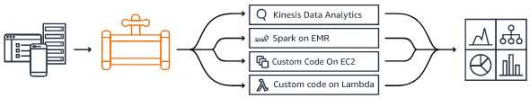
Press Get started.

Get started with Amazon Kinesis

To get started, choose an Amazon Kinesis resource to create.

Ingest and process streaming data with Kinesis streams

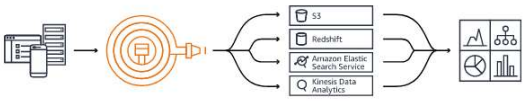
Process data with your own applications, or using AWS managed services like Amazon Kinesis Data Firehose, Amazon Kinesis Data Analytics, or AWS Lambda.



Create data stream

Deliver streaming data with Kinesis Firehose delivery streams

Continously collect, transform, and load streaming data into destinations such as Amazon S3 and Amazon Redshift.



Create delivery stream

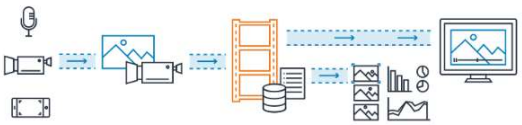
Analyze streaming data with Kinesis analytics applications

Run continous SQL queries on streaming data from Kinesis data streams and Kinesis Firehose delivery streams.

Create analytics application

Ingest and process media streams with Kinesis video streams

Build applications to process or analyze streaming media.



Create video stream

Related CloudFormation templates

Choose an AWS CloudFormation template to get started building a complete streaming data solution. AWS CloudFormation templates offer quick and reliabl provisioning of services and applications. Learn more

CloudFormation template	Description	Recommended for
Real-time insights on AWS account activity	Gain valuable insights on your AWS usage in seconds with a real-time dashboard. No coding required. Learn more	AWS customers with prior usage data
AWS connected vehicle	Implement secure vehicle connectivity services in the AWS Cloud, including local computing in vehicles, event rules, data processing, and storage. Learn more	Advanced AWS IoT customers

Cancel

If there are already streams or applications created, the Amazon Kinesis Dashboard looks like:

Amazon Kinesis dashboard

Amazon Kinesis makes it easy to collect, process, and analyze video and data streams in real time, so you can get timely insights and react quickly to new information. What is streaming data?

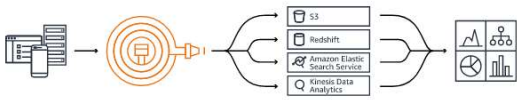
Kinesis data streams (2)

Name	Status
DEMO_LAMBDA1	Active
kinesis-analytics-demo-stream	Active

View all | Create data stream

Kinesis Firehose delivery streams

Continously collect, transform, and load streaming data into destinations such as Amazon S3 and Amazon Redshift.
Learn more



Create delivery stream

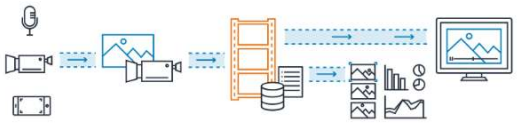
Kinesis analytics applications (2)

Name	Status
ExampleApp	Ready
demo-lambda-1	Ready

View all | Create analytics application

Kinesis video streams

Build applications to process or analyze streaming media.
Learn more



Create video stream

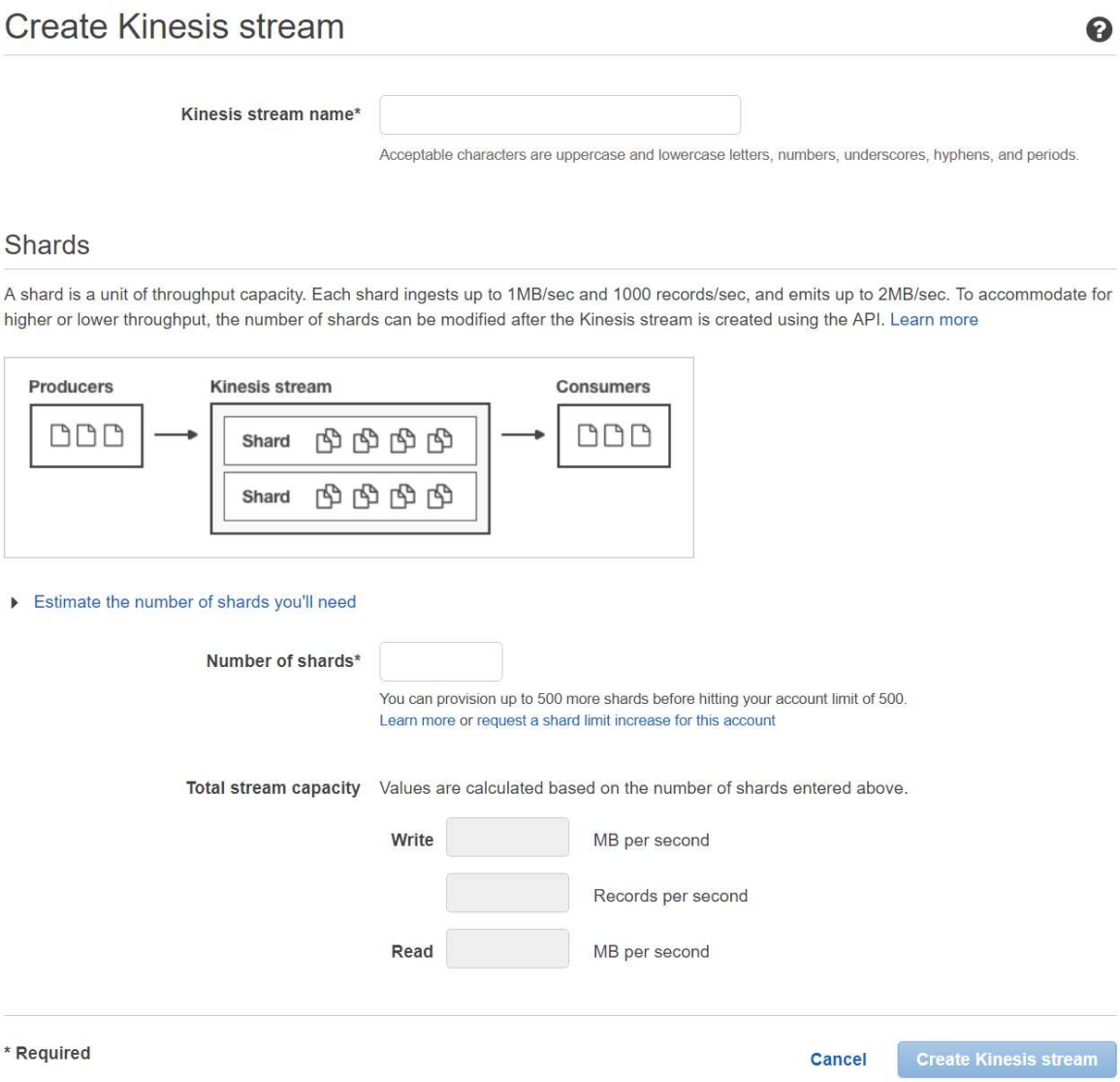
Related CloudFormation templates

Choose an AWS CloudFormation template to get started building a complete streaming data solution. AWS CloudFormation templates offer quick and reliable provisioning of services and applications. Learn more

CloudFormation template	Description	Recommended for
Real-time insights on AWS account activity	Gain valuable insights on your AWS usage in seconds with a real-time dashboard. No coding required. Learn more	AWS customers with prior usage data
AWS connected vehicle	Implement secure vehicle connectivity services in the AWS Cloud, including local computing in vehicles, event rules, data processing, and storage. Learn more	Advanced AWS IoT customers

Kinesis Datastreams

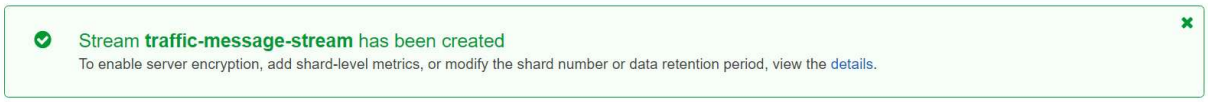
On the Kinesis dashboard, choose Create datastream.



Note: where you see {your alias}, enter an alias to distinguish the different objects created by the different users!

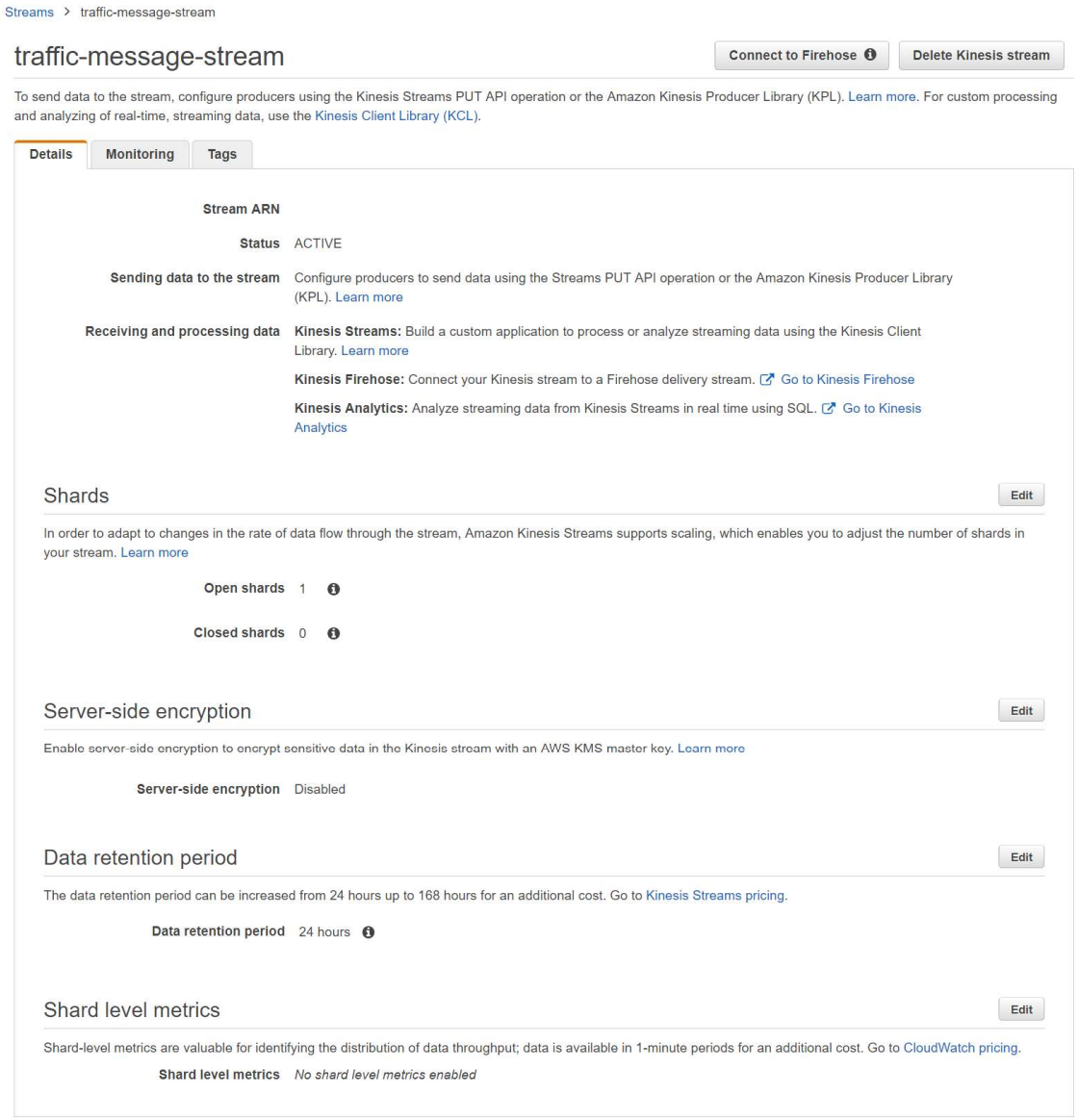
Give the stream a name: **{your alias}-traffic-message-stream**
Number of shards: **1**
The total stream capacity is shown.
Press Create Kinesis stream
The stream is now being created.

When ready (it will not take long), a message is shown that the stream has been created:



Press on the created stream in the table:

The details page is shown.

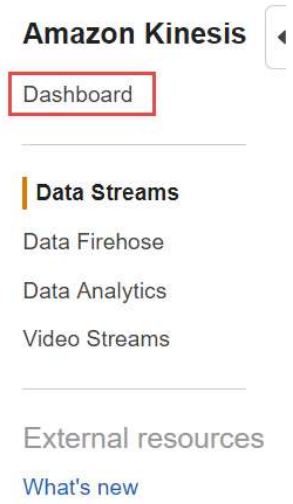


Now we have a datastream on which data can be received.
On the monitoring tab, you see several metrics about the datastream. But currently we do not have any data yet, so nothing is shown.

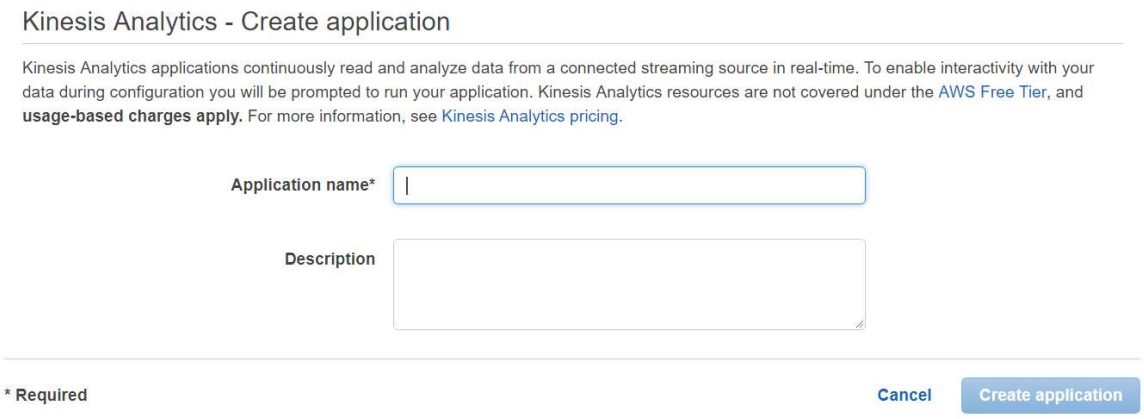
Kinesis Analytics

Create analytics application

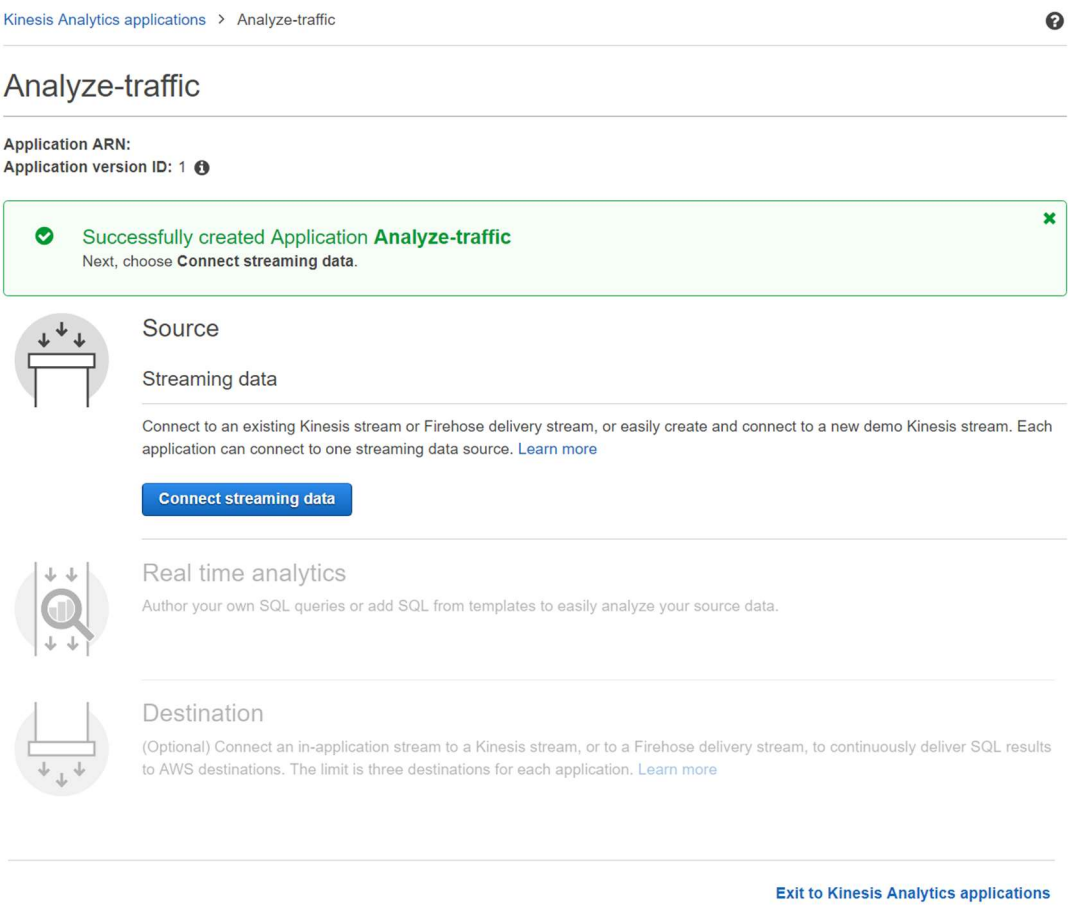
Go back to the Dashboard.



Now create an analytics application so we can do something with the data.
Press Create analytics application.

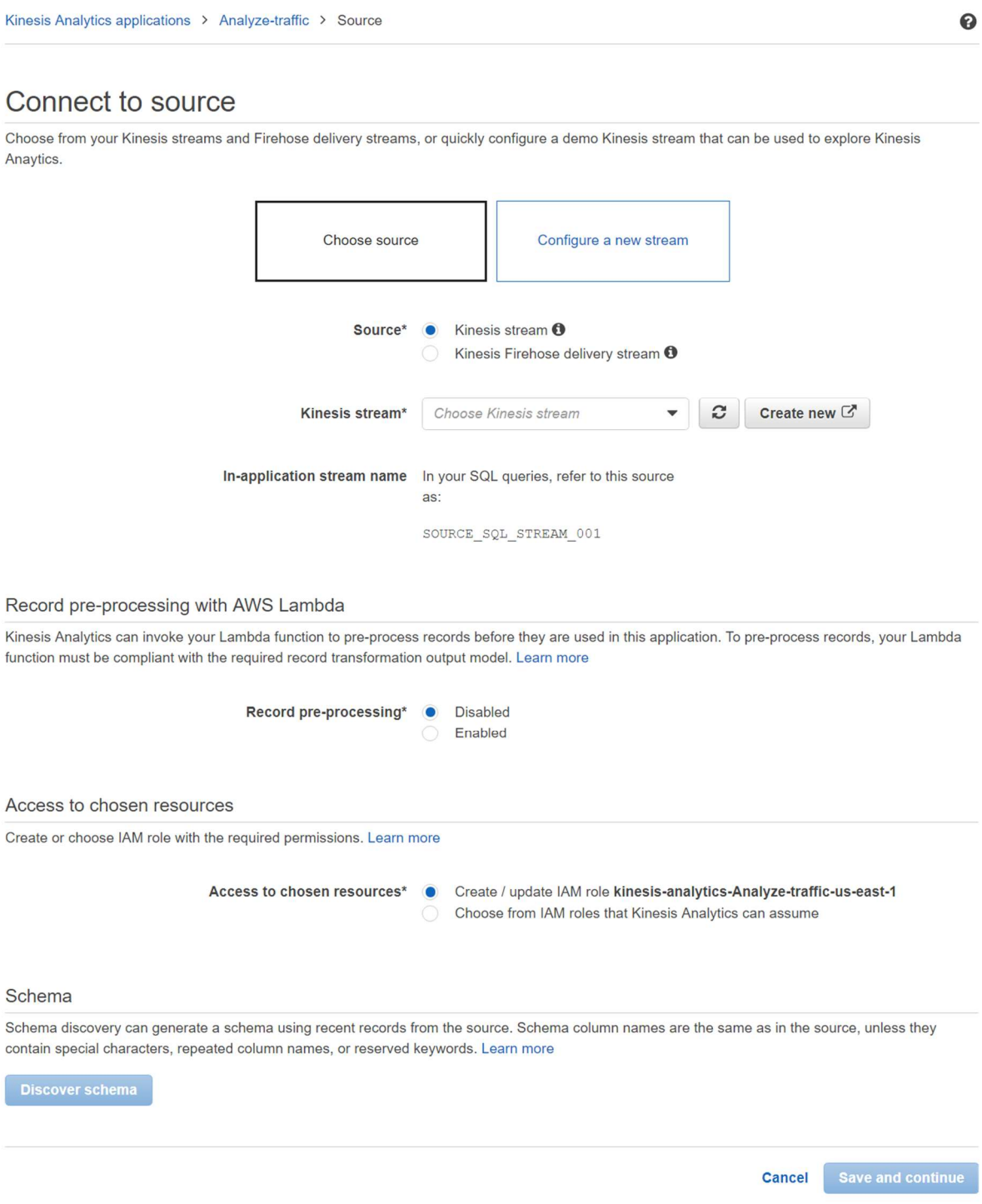


Give the application a name: **{your alias}-Analyze-traffic**
And optional a description.
Press Create application.

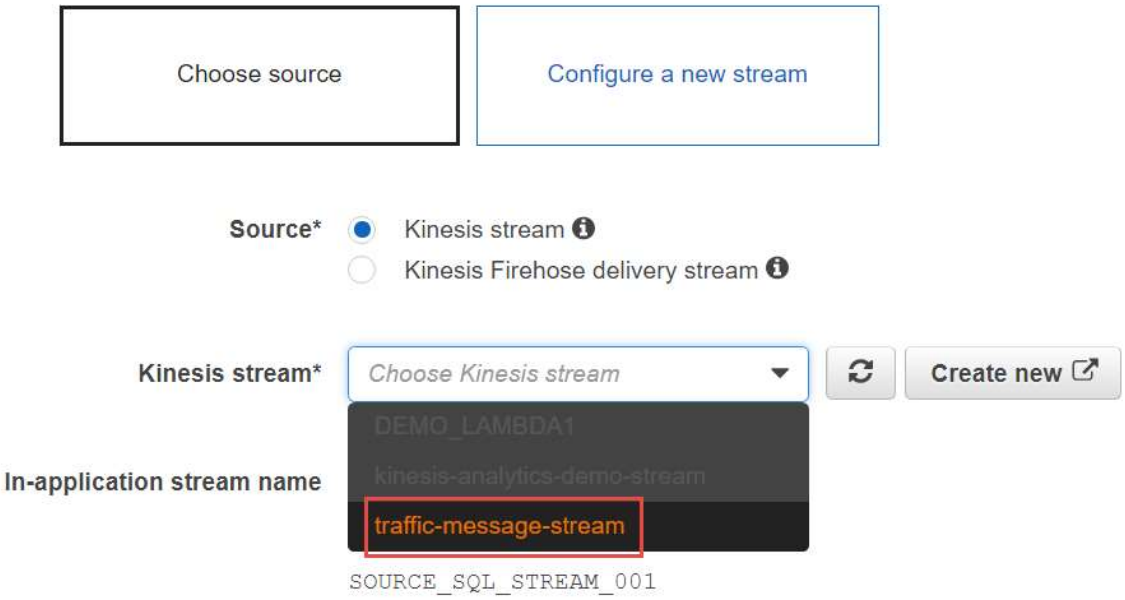


Connect streaming data

To be able to do anything, we first need to add the source.
Press connect streaming data.



We already created a stream before, so select this one from the drop down list.



To be able understand the message coming in, we need to set the schema of the data stream.

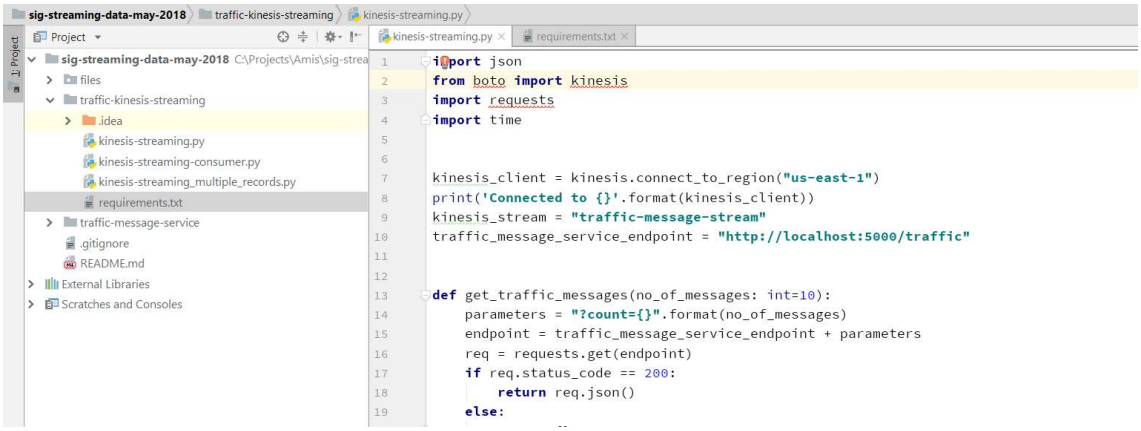
Data producing

But remember: we did not have any data yet in the stream, so there discovering will fail.
We now first need to send data to the created datastream.
We will use a client on the local pc to send the data to the Kinesis datastream.

Before we can do this, we first need to install the client sample code.

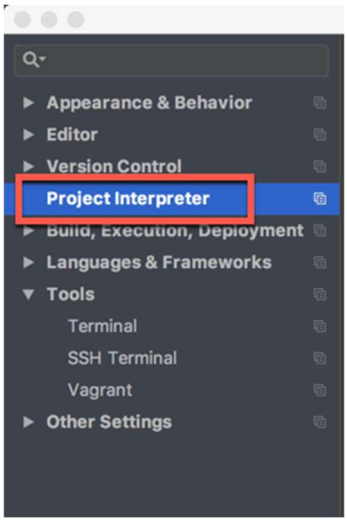
Clone or download the following repository:
https://github.com/AMIS-Services/sig-streaming-data-may-2018

This repository contains a simple client which can sent data to Kinesis inside of the traffic-kinesis-streaming subfolder (https://github.com/AMIS-Services/sig-streaming-data-may-2018/tree/master/traffic-kinesis-streaming).
Open the whole project in PyCharm to see the code.

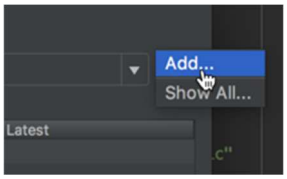


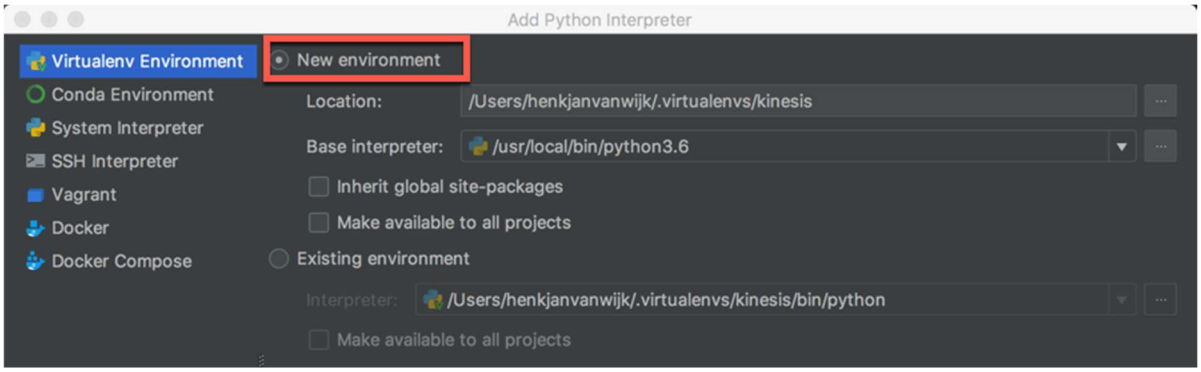
If you see the red underlining, this means that the required Python modules are not yet available. First a virtualenv must be created (or selected).

To run to Python script, first create a Python virtualenv (if you did not do that already):
In the menu select File > Default Settings.
Navigate to Project Interpreter.



In the right pane, click on the wheel icon 
Select Add in the popup menu.

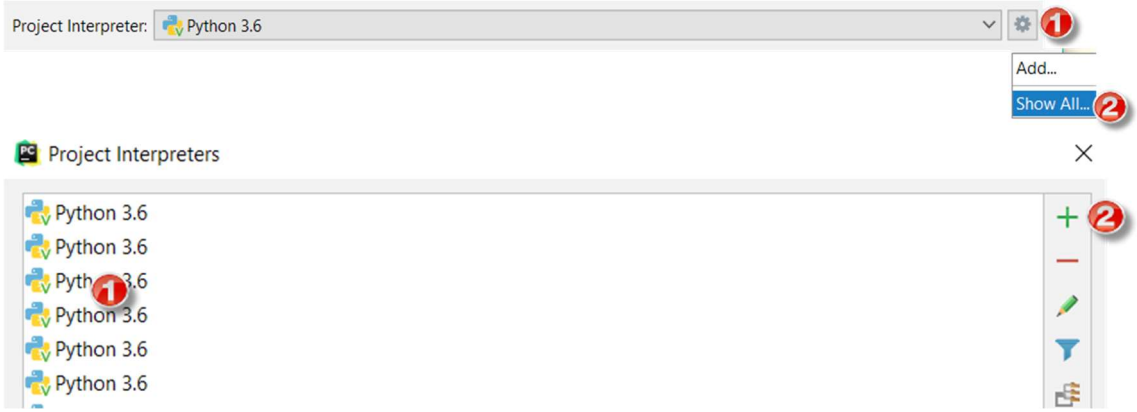




Select Virtualenv Environment, New environment and enter a location, eg. **kinesis** or **kinesis-workshop**.
The last folder (in the screenshot: kinesis) is the name of the virtual environment.
Press OK.

A new virtual environment is created and linked to the project.

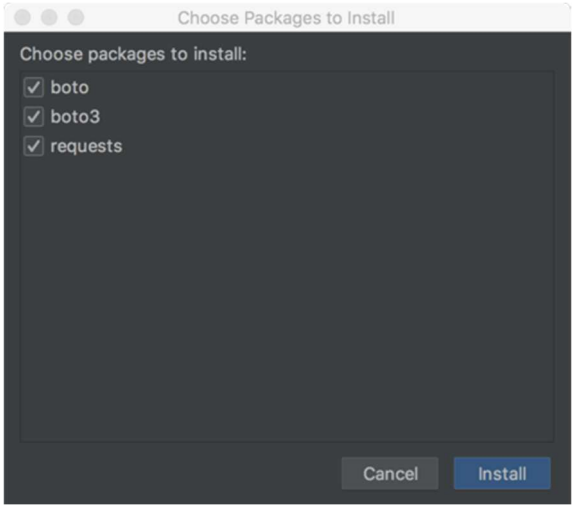
If you already created a virtualenv, then you need to select the already created virtualenv:



Select an existing virtualenv or add one to the list using the plus icon.

You will get an error that certain package requirements are not satisfied.

Press install requirements.



Choose all packages (the default) and press Install.

The packages are now being downloaded and installed (using pip).

In the script the variable traffic_message_service_endpoint contains the url to the traffic message API.
Use either:

- localhost:8080/traffic
- http://traffic-message-service.s7whvi2kjr.us-east-1.elasticbeanstalk.com/traffic

The first you can use when running the API locally or the second one using an already installed API on AWS Elastic Beanstalk.

Running the local REST API service

In the folder traffic-message-service you will find the (simple) implementation of the REST API service for generating traffic containing random license plates.
This API is implemented using Python Flask.
To be able to run this API locally, you can either run the script from a command line or using PyCharm.

Command line

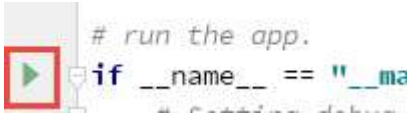
Open a command prompt and navigate to the traffic-message-service folder.
Activate the virtualenv and start the server: `python application.py`

```
(traffic-message-service) PS C:\Projects\Amis\sig-streaming-data-may-2018\traffic-message-service> python application.py
* Restarting with stat
* Debugger is active!
* Debugger PIN: 231-292-656
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

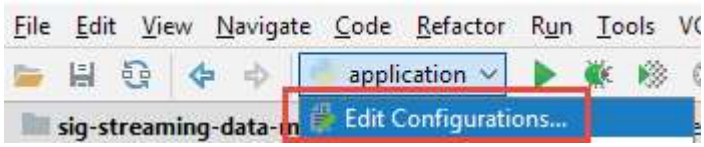
The used port is being displayed.

Using PyCharm

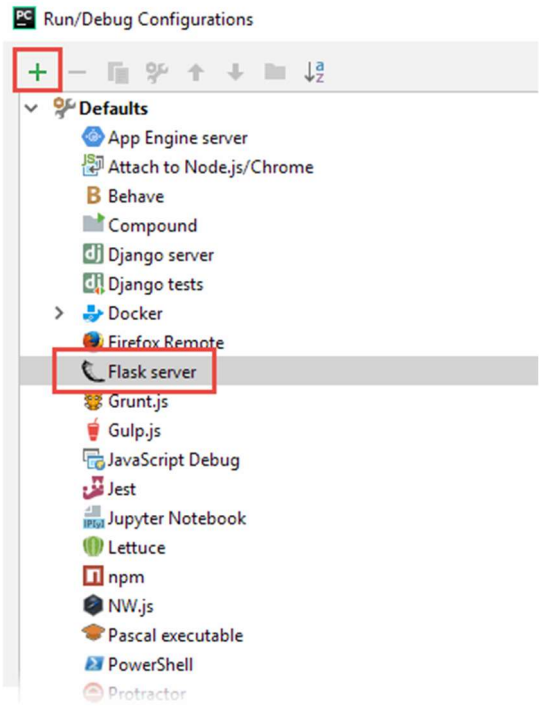
Open application.py and press the run icon in the margin of the editor



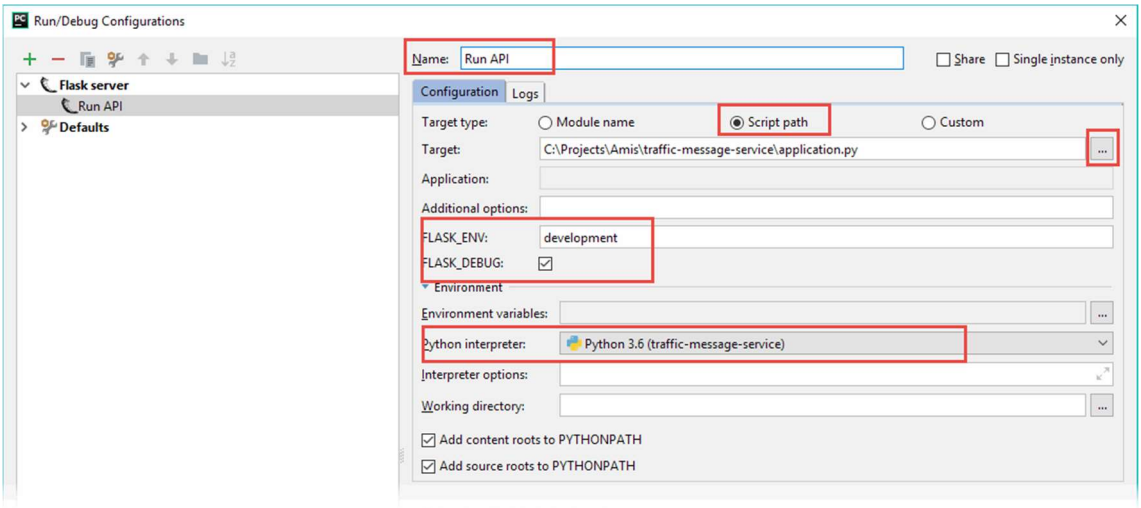
Or add a Run configuration.



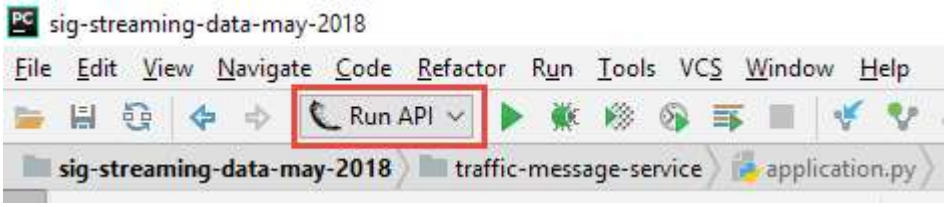
Press the plus icon and choose Flask Server



Enter a name, choose Script path and select the application.py file.
Make sure the correct virtualenv is selected under Python interpreter.
And press OK.



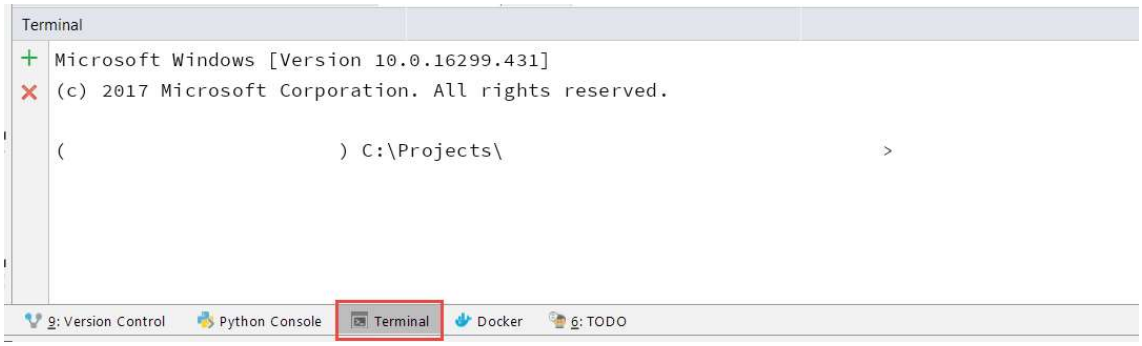
Now the option is added to the drop down list next to the Run button.



Press the Run button to start the API.

Toubleshooting

If you do not get the message for installing the missing modules, you can also do it by hand in a terminal window.
Press the terminal icon in the footerbar.



Between the brackets in the terminal window you should see the name of your virtualenv. This means the virtualenv is activated.

From the rootfolder of the repository type:

```
pip install --upgrade -r traffic-kinesis-streaming\requirements.txt
pip install --upgrade -r traffic-message-service\requirements.txt
```

Run streaming producer

Press the small run button next to `if __name__ == "__main__":`

The client will now be run.

Take a look at the code. You will notice that the client is sending the data one record at a time.This can be done more efficiently (if you do not need the messages to be sent one at a time), by combining them in a multi record list.

Open the file '*kinesis_streaming_multiple_records.py*' to see the example for this type of message sending. If you look carefully at the difference between the 2 scripts, you will see that the single record script uses **put_record** and the multi record script uses **put_records.** And that is not the only difference. The first script uses the boto library and the second one the boto3 library. Both libraries are maintained by Amazon. Boto3 is the newest. Boto is more legacy code andhas a class for each of the AWS services. Boto3 is more generic.

```
kinesis_client = kinesis.connect_to_region("us-east-1")
```

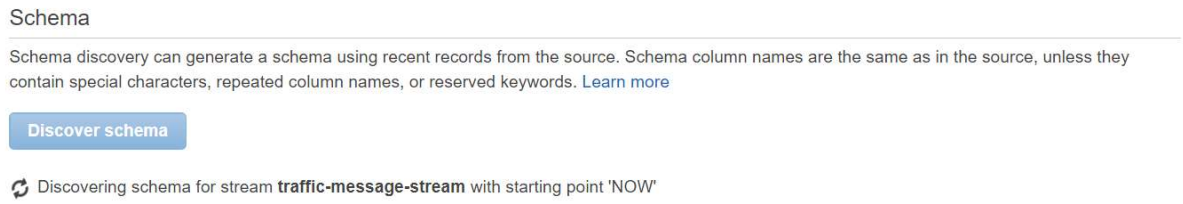
vs

```
kinesis_client = boto3.client('kinesis')
```

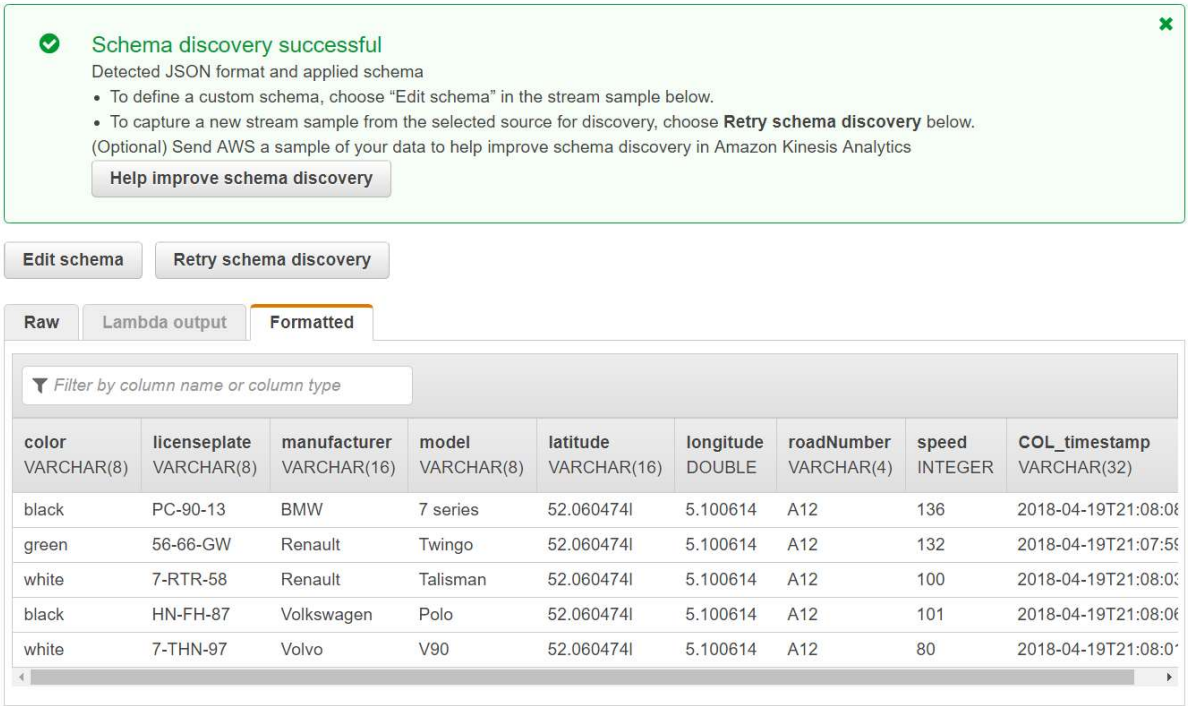
After starting the client, data will be coming into the Kinesis stream.

Discover schema

Go back to the AWS page you left.
Scroll down to Schema.
Press Discover schema.



Wait for a while until the schema has been analyzed from the incoming data.



Check if the schema is OK.
Press Edit schema and make any necessary changes, such as datatype and data length.
After you made the necessary changes, save the changes.

The stream is now checked against the updated schema.
After a while new rows will be shown (if the schema is ok).

When you are done, press Exit (done).

Press 

After successful discovery of the schema, stop the client of producing data (to minimize the running costs of using Kinesis).
Start the client again when you need some new data.
Also stop a running Kinesis Analytics Application if you do not need it.

Stopping an Kinesis Analytics Application

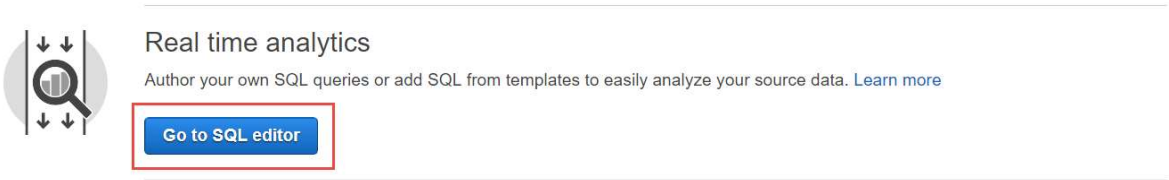
You can only create and delete an application in the console. You need to use the AWS CLI to stop a running application:

```
aws --region us-east-1 kinesisanalytics stop-application --application-name
Analyze-traffic
```

Enter the correct region and application name.

Performing some analytics

Now we can do some analysis.
Press Go to SQL editor.



Simple streaming

Let start with a ‘simple’ query to show the streaming data.

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
carname varchar(20),
carmodel varchar(20),
licenseplate varchar(10),
country varchar(20));

CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO
"DESTINATION_SQL_STREAM"
SELECT STREAM "manufacturer", "model", "licenseplate", "country"
FROM "SOURCE_SQL_STREAM_001"
```

All real time analytics queries look like this one.
We start with creation a destination stream (kind of table) in which you define the column the stream will have and its datatypes.
Next you define a pump which will take data from the source data stream you created earlier and insert the data into then destination stream.
The pump and the stream must match of course.

Select aggregated data (Using GROUP BY)

A more advanced query is the one we use to answer the following question:

- We are interested in the total number of cars grouped by manufacturer within a one-minute window.

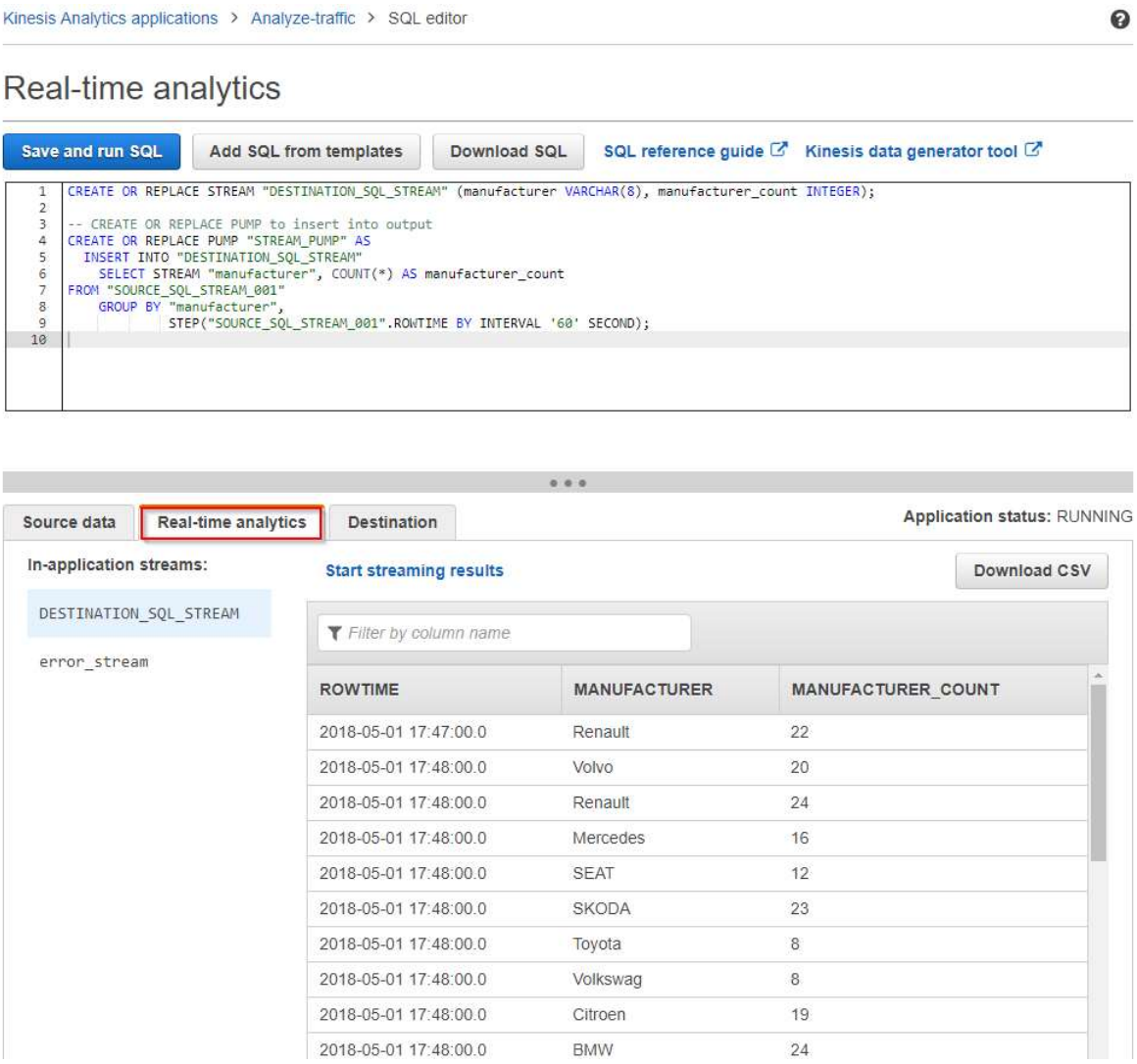
The query below answers our question:

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (manufacturer
VARCHAR(8), manufacturer_count INTEGER);

-- CREATE OR REPLACE PUMP to insert into output
CREATE OR REPLACE PUMP "STREAM_PUMP" AS
  INSERT INTO "DESTINATION_SQL_STREAM"
    SELECT STREAM "manufacturer", COUNT(*) AS manufacturer_count
FROM "SOURCE_SQL_STREAM_001"
    GROUP BY "manufacturer",
             STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '60'
SECOND);
```

Select the query above paste it in *SQL editor and* press the *Save and run SQL button*

The result could be something like below:

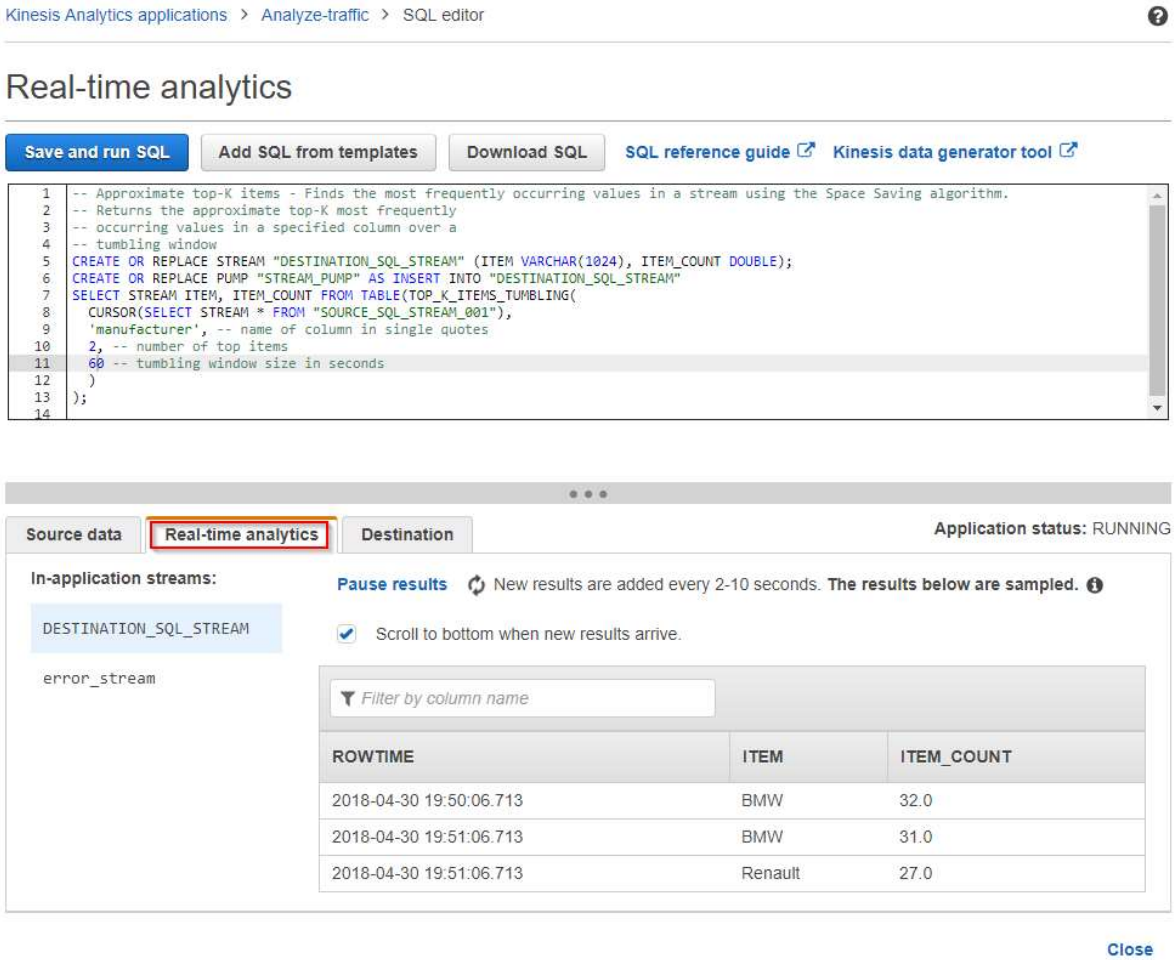

Kinesis Analytics applications > Analyze-traffic > SQL editor
Real-time analytics
Save and run SQL
Add SQL from templates
Download SQL
SQL reference guide
Kinesis data generator tool
1 CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (manufacturer VARCHAR(8), manufacturer_count INTEGER);
2
3 -- CREATE OR REPLACE PUMP to insert into output
4 CREATE OR REPLACE PUMP "STREAM_PUMP" AS
5 INSERT INTO "DESTINATION_SQL_STREAM"
6 SELECT STREAM "manufacturer", COUNT(*) AS manufacturer_count
7 FROM "SOURCE_SQL_STREAM_001"
8 GROUP BY "manufacturer",
9 STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '60' SECOND);
10
Source data
Real-time analytics
Destination
Application status: RUNNING
In-application streams:
DESTINATION_SQL_STREAM
error_stream
Start streaming results
Download CSV
Filter by column name
ROWTIME
MANUFACTURER
MANUFACTURER_COUNT
2018-05-01 17:47:00.0 Renault 22
2018-05-01 17:48:00.0 Volvo 20
2018-05-01 17:48:00.0 Renault 24
2018-05-01 17:48:00.0 Mercedes 16
2018-05-01 17:48:00.0 SEAT 12
2018-05-01 17:48:00.0 SKODA 23
2018-05-01 17:48:00.0 Toyota 8
2018-05-01 17:48:00.0 Volkswag 8
2018-05-01 17:48:00.0 Citroen 19
2018-05-01 17:48:00.0 BMW 24

Select Most Frequently Occurring Values (top-K items)

In our message we have a car manufacturer. Suppose we want to know what the two most occurring car manufacturers are within a specific time window(tumbling).
From the *SQL editor* window select the *Add SQL from templates* button and choose the *Approximate top-K items template*. Select the *Use this SQL instead* button (this will overwrite your previous query!.).
Change the query so it looks like the query below:

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (ITEM VARCHAR(1024),
ITEM_COUNT DOUBLE);
CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO
"DESTINATION_SQL_STREAM"
SELECT STREAM ITEM, ITEM_COUNT FROM TABLE(TOP_K_ITEMS_TUMBLING(
  CURSOR(SELECT STREAM * FROM "SOURCE_SQL_STREAM_001"),
  'manufacturer', -- name of column in single quotes
  2, -- number of top items
  60 -- tumbling window size in seconds
  )
);
```

NOTE: The bold text shows the changes you have to make.
If the changes are made, press the *Save and run SQL button*. The result could be something like below:



Based on the testdata we use, the result above could be correct. The manufacturers that are shown belong to the two most frequently occurring car manufacturers.
Further explanation about tumbling windows see documentation:

- https://docs.aws.amazon.com/kinesisanalytics/latest/dev/tumbling-window-concepts.html
- Tumbling windows are even better explained in the Microsoft documentation: https://docs.microsoft.com/nl-nl/azure/stream-analytics/stream-analytics-window-functions

Reference data

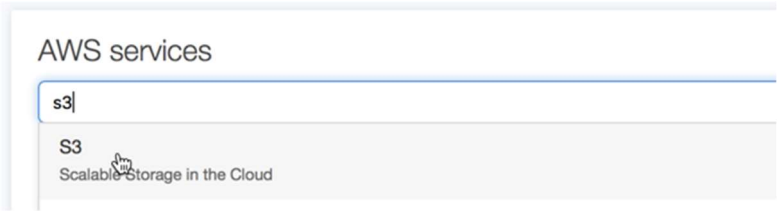
In our message we have a country code, but suppose you want to use the country name instead.
We can do this by adding a reference table and perform a join.
As a reference table we will use a csv file with the necessary translation.
This csv file must be uploaded to a S3 bucket and given access to Kinesis.

We prepared the S3 bucket already for you. You will only need to add the reference file.

But if you want to know how the S3 bucket is created, see the following paragraph, otherwise skip to “Adding file to S3 bucket” (page 36).

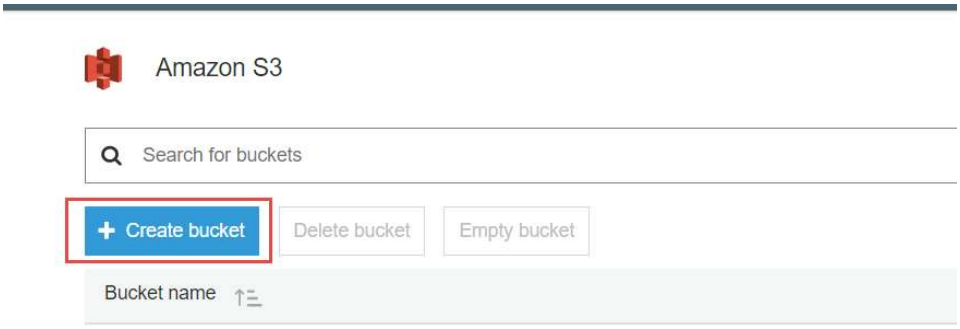
Create a S3 bucket

Go to the AWS console homepage and type s3 in the search box.

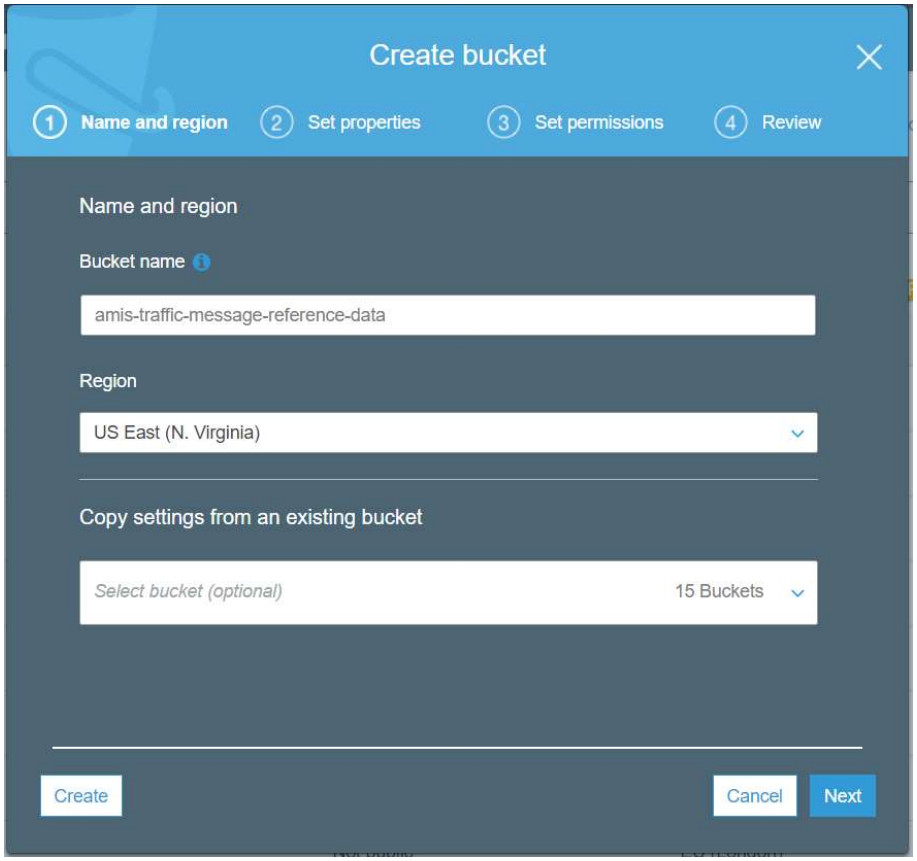


Select S3.

Press Create Bucket.

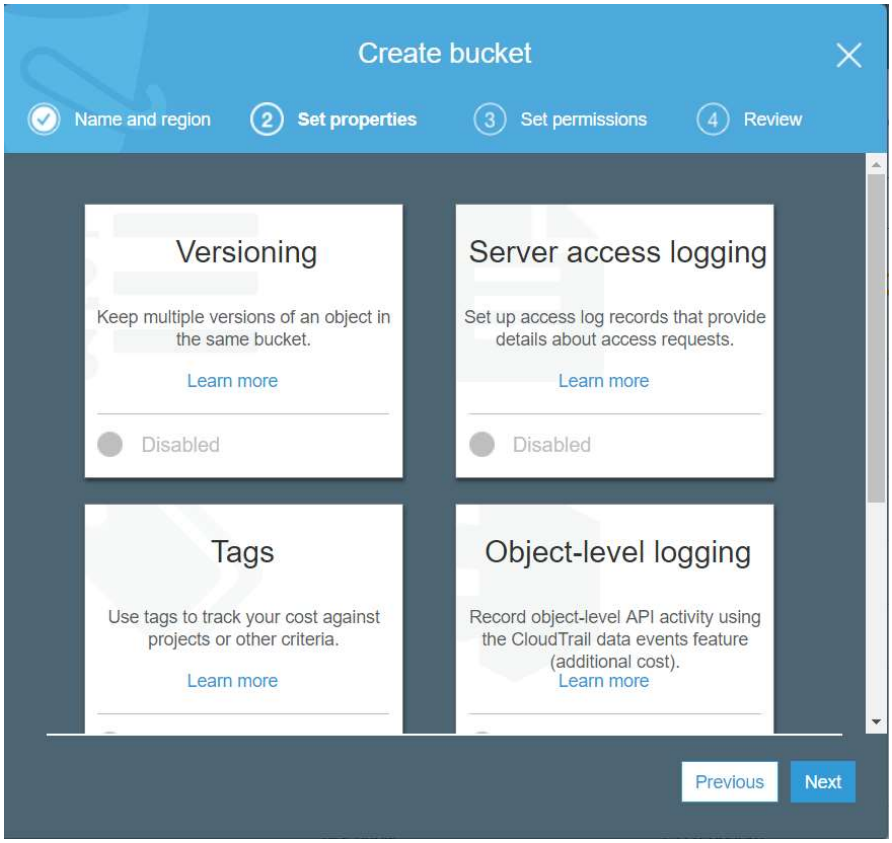


traffic-message-reference-data

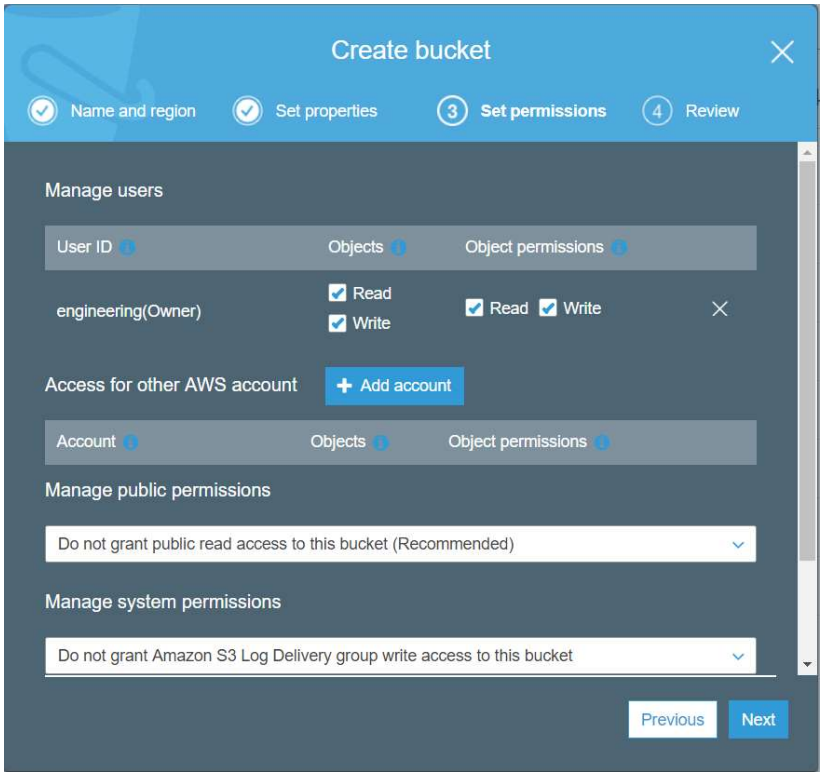


Give the bucket a name: **amis-traffic-message-reference-data**
Select a Region: US East (N. Virginia)

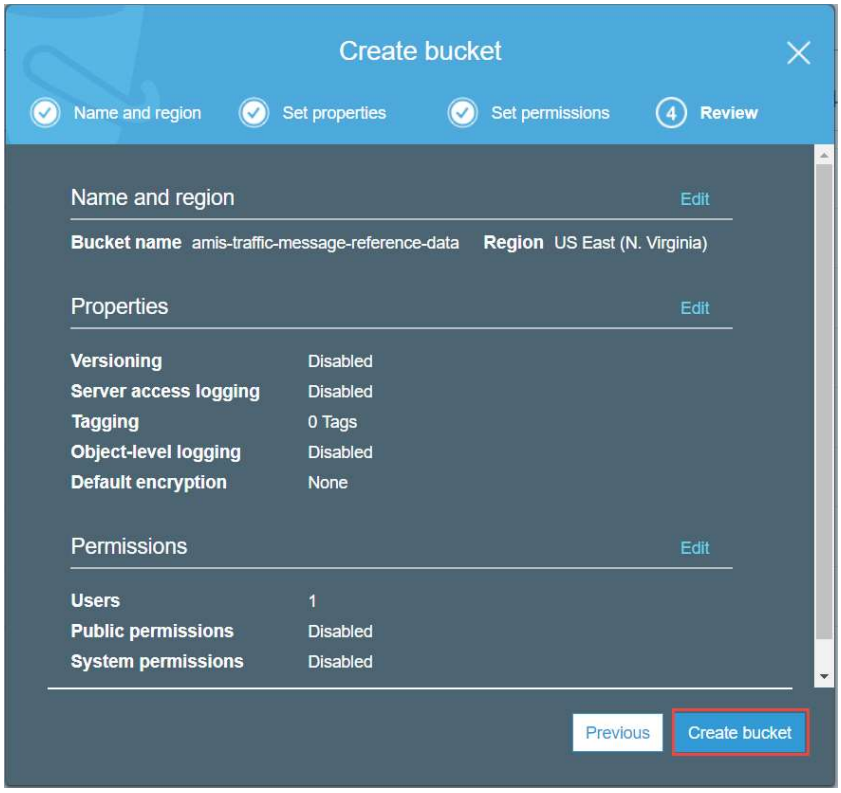
Press Next



No properties are needed, so press Next again.



On the permission tab, leave the default options.
No public permissions.
Press Next.



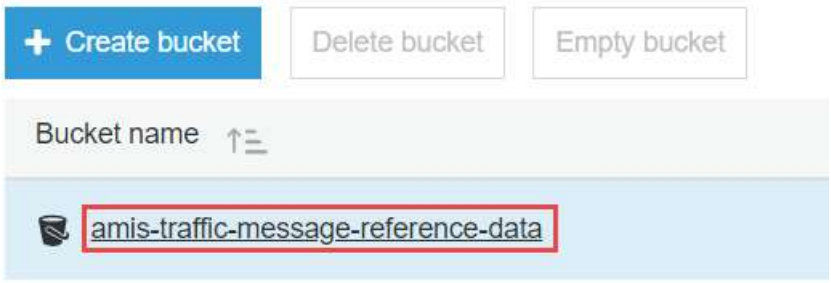
Press Create bucket to create the bucket.

Adding file to S3 bucket

Navigate to the S3 service (if you are not there already after creating the bucket).

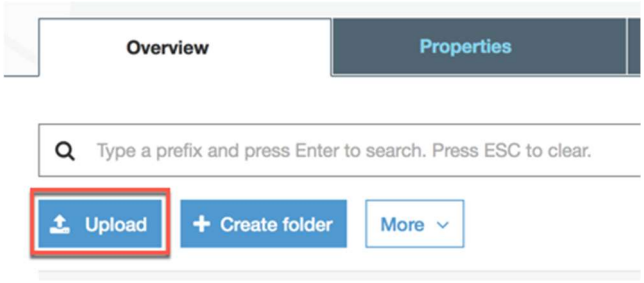
Find the bucket.

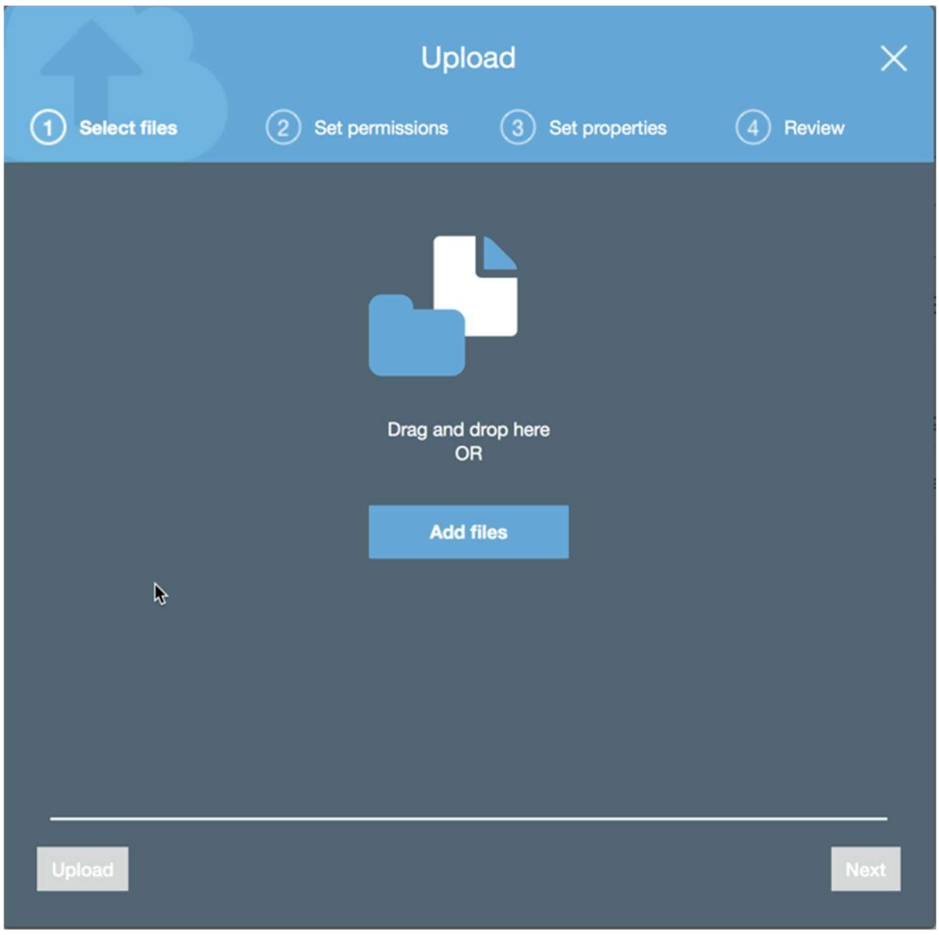
Hover over the row and click on the link (notice the underlining!)



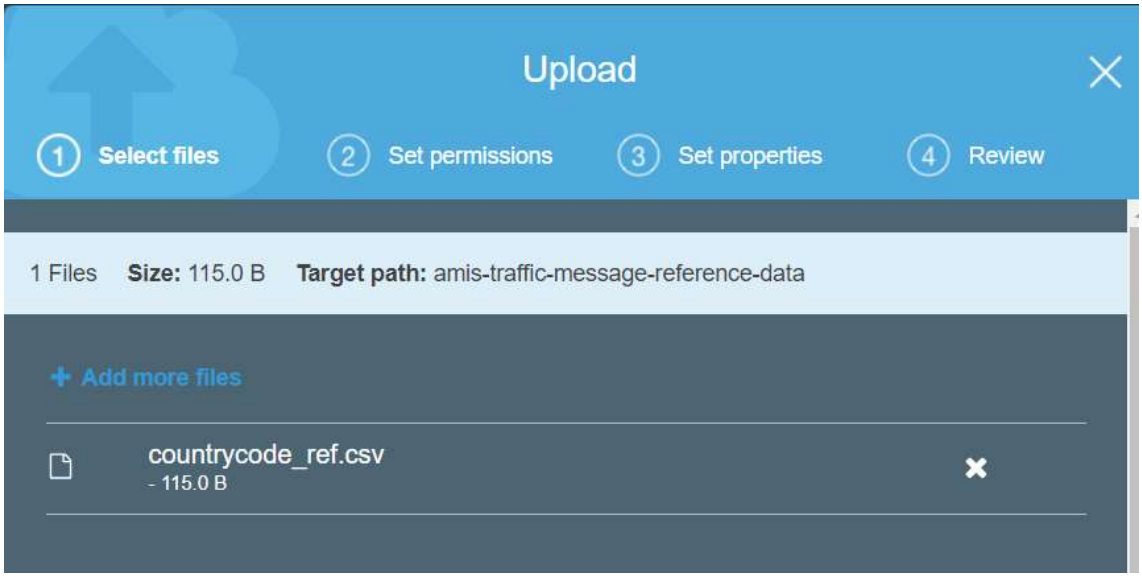
Clicking on the row only will show the properties of the bucket, not the contents.

Press the Upload button.



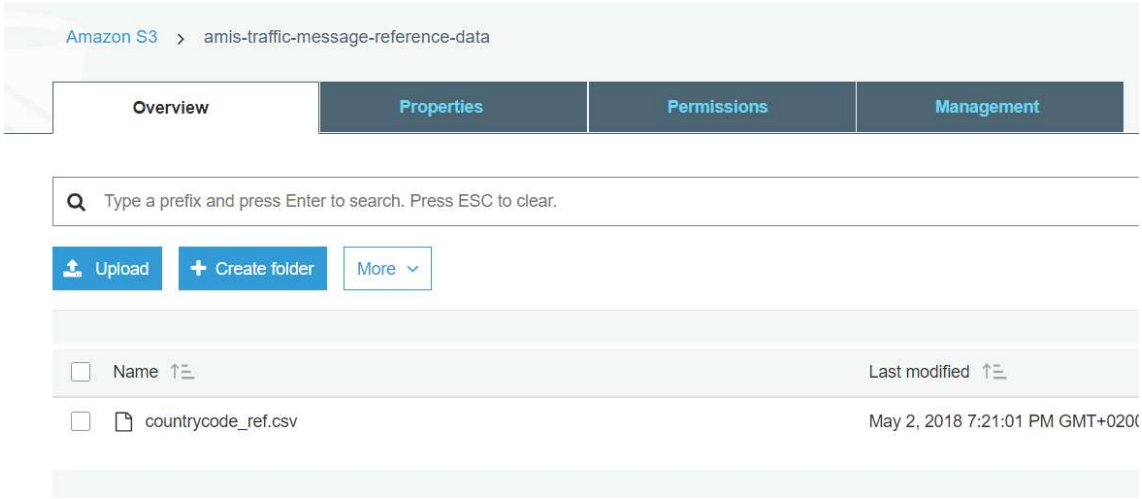


Copy the file *'countrycode_ref.csv'* from the share to your local PC and rename the file to ***{your alias}_countrycode_ref.csv***. And drop the renamed file on the window or use the Add files button.



Press Upload.

The file will appear in the table.



Unfortunately it is not possible to set the permission for accessing this file from the Kinesis application using the console. We need to use the AWS CLI for this.

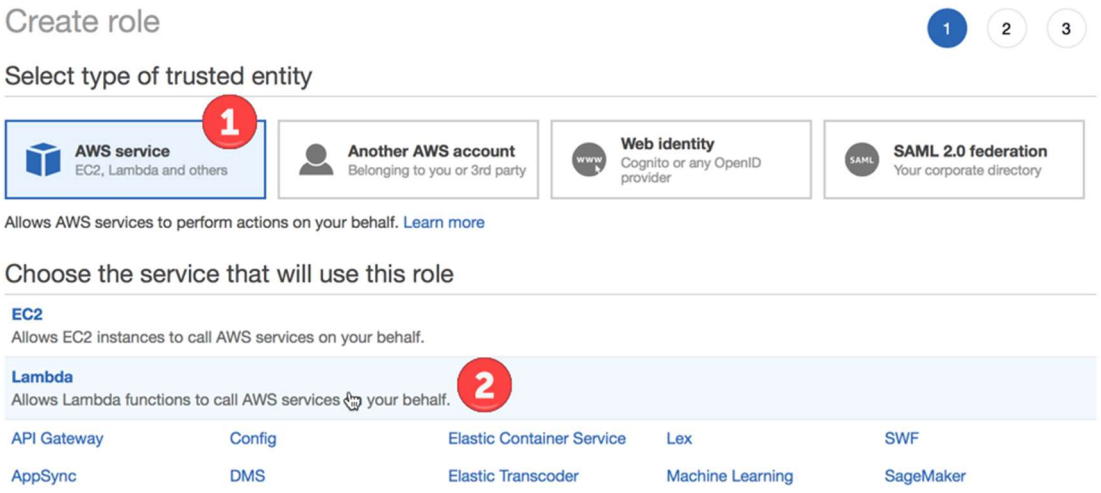
IAM role

But first we need a new IAM role called **KinesisAnalytics-ReadS3Object** if this not already created. We did this already for you.

But if you want to know the steps involved, then see the following paragraph. Otherwise skip to paragraph “Add Reference Data Source to the Application Configuration” on page 41.

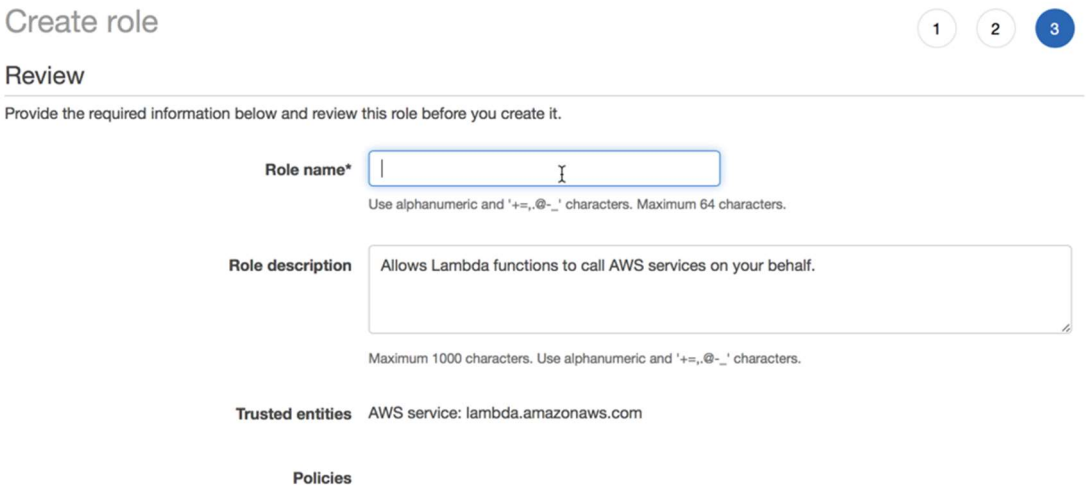
Create a new IAM role

Go to the IAM page.
Select Roles.
Press Create Role.



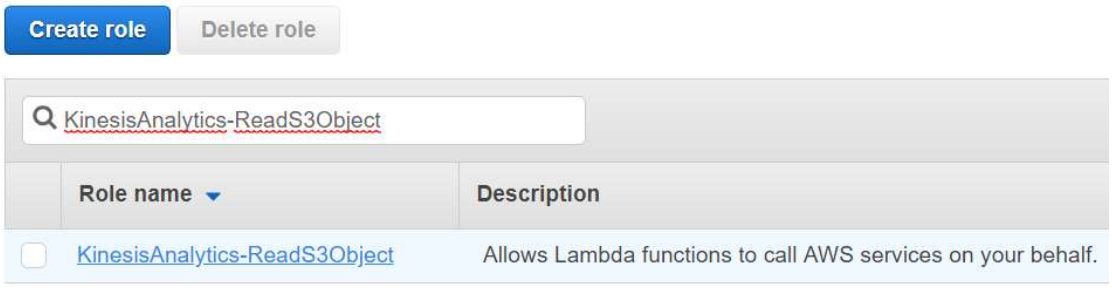
Select AWS service.
Select Lambda.

Press Next: Permissions
Do not add any policies.
Press Next: Review

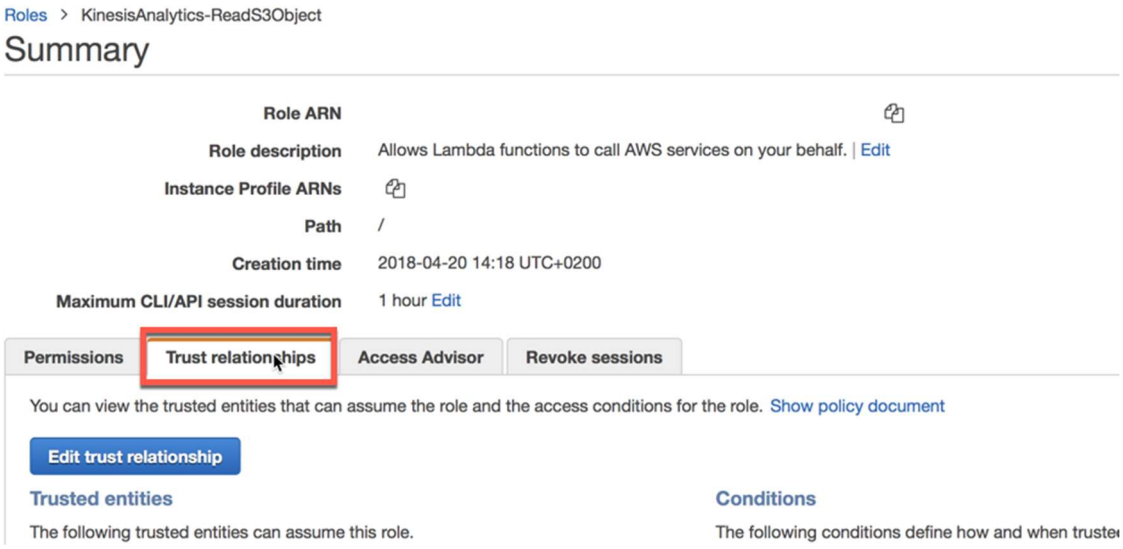


Enter the Role name: **KinesisAnalytics-ReadS3Object**
Press Create role

After creating the role, select the role you just created.



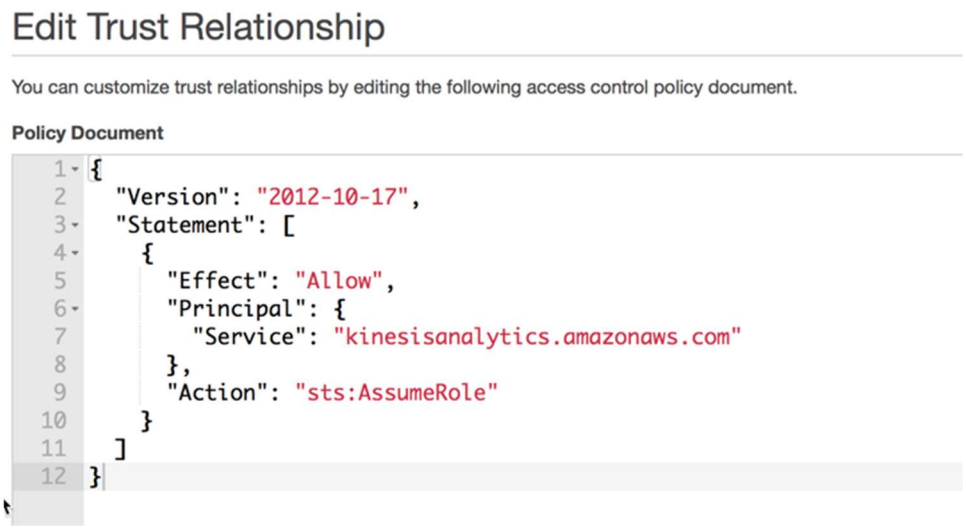
Go to the Trust relationships tab



Press Edit trust relationship.

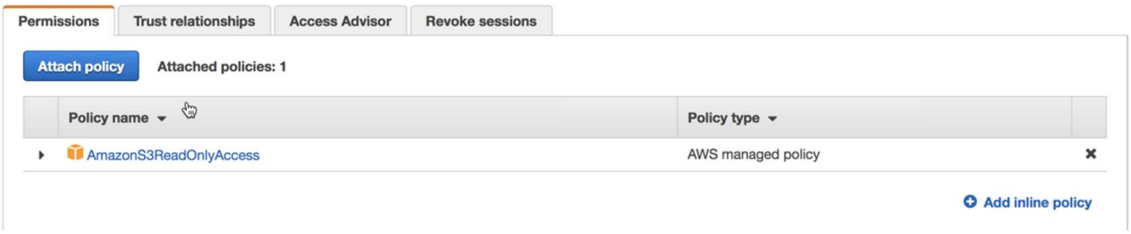
Enter:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```



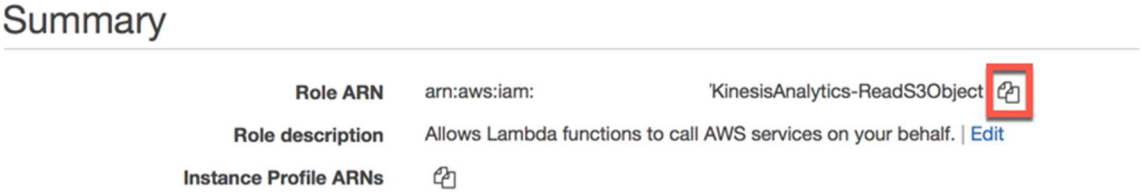
Press Update Trust Policy

Go to the Permissions tab and attach an AWS managed policy called **AmazonS3ReadOnlyAccess**.
This grants the role permissions to read an S3 object.



Press attach policy.

Add Reference Data Source to the Application Configuration
Navigate to the IAM Roles and find the role KinesisAnalytics-ReadS3Object.
The select the role and copy the Role ARN.



Save it somewhere. You will need it in a minute.

Now we are going to add the reference data source to the Kinesis Analytics Application.
We will use the AWS CLI.

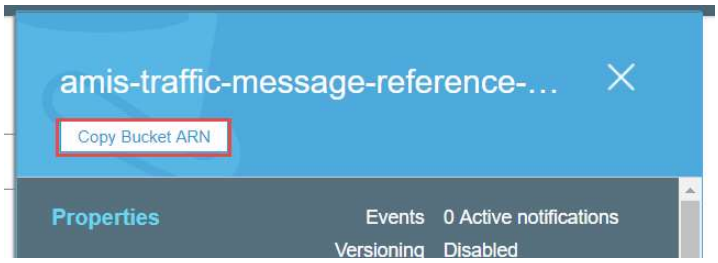
Open a command prompt or Powershell window.

Run the describe-application command to get the application details.

```
aws kinesisanalytics describe-application --region us-east-1 --application-name application-name
```

Replace application-name with the name of the Kinesis Analytics Application.
Note the application version Id. We will need that in a minute.

You also need the S3 bucket ARN.
Go to the S3 console, select the bucket (not the link!).
The properties are shown.
Press Copy Bucket ARN.



Save it somewhere.

On the command prompt / shell, type:

```
aws kinesisanalytics add-application-reference-data-source --endpoint https://kinesisanalytics.us-east-1.amazonaws.com --region us-east-1 --application-name traffic-stream-demo --reference-data-source '{\"TableName\":\"Countries\",\"S3ReferenceDataSource\":{\"BucketARN\":\"arn:aws:s3:::traffic-message-reference-data\",\"FileKey\":\"countrycode_ref.csv\",\"ReferenceRoleARN\":\"arn:aws:iam::######:role/KinesisAnalytics-ReadS3Object\"},\"ReferenceSchema\":{\"RecordFormat\":{\"RecordFormatType\":\"CSV\",\"MappingParameters\":{\"CSVMappingParameters\":{\"RecordRowDelimiter\":\"\n\",\"RecordColumnDelimiter\":\",\"}}},\"RecordEncoding\":\"UTF-8\",\"RecordColumns\":[{\"Name\":\"countrycode\",\"SqlType\":\"VARCHAR(5)\"},{\"Name\":\"countryname\",\"SqlType\":\"VARCHAR(64)\"}]}}' --current-application-version-id 7
```

Note the red marked entries. Replace them with the your own values.
Note the blue marked entries. These are the table name as it will be known in the Kinesis Analytics Application and the used filename.

Using the reference data

Create a new Kinesis Analytics Applications or just replace one.

Enter the following query:

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (carname varchar(20),
carmodel varchar(20),
licenseplate varchar(10),
country varchar(20)
);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
SELECT STREAM "manufacturer", "model", "licenseplate", "c"."countryname"
FROM "SOURCE_SQL_STREAM_001" left join "Countries" as "c"
ON "SOURCE_SQL_STREAM_001"."country" = "c"."countrycode"
```

This is similar to the first simple streaming query, but now with the country name instead of the code.

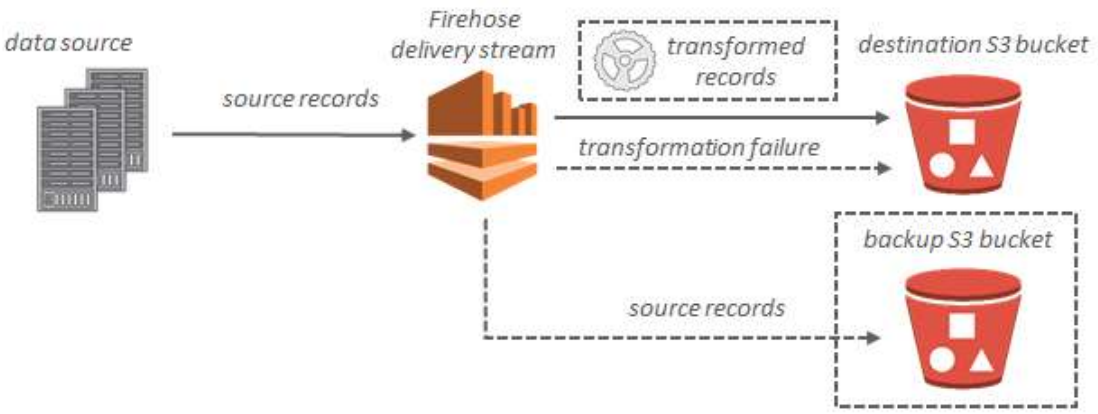
Offloading data

Offloading data in Kinesis can be done by adding a Kinesis Firehose delivery stream. Firehose can offload data to S3 buckets, Redshift, Amazon Elastic Search and to another Kinesis Analytics Application.

Dataflow

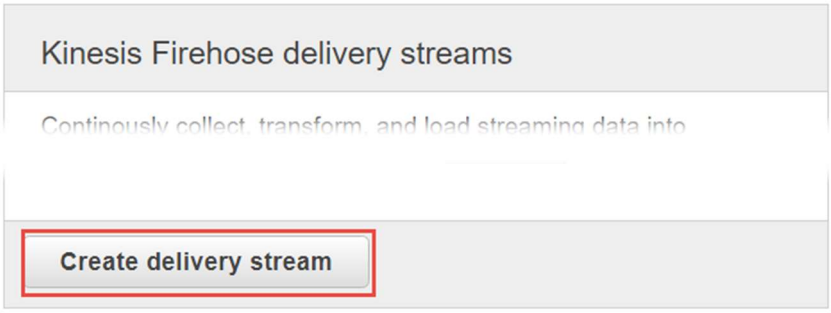
Offload to S3

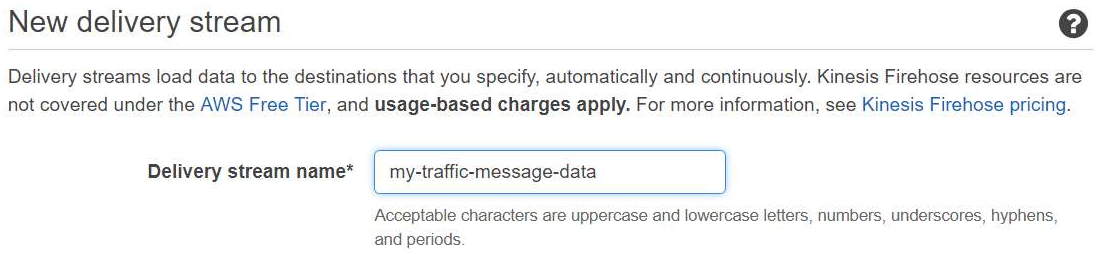
For Amazon S3 destinations, streaming data is delivered to your S3 bucket. If data transformation is enabled, you can optionally back up source data to another Amazon S3 bucket.



Let's start with creating a new Firehose Delivery stream.

Press Create delivery stream on the Kinesis dashboard.





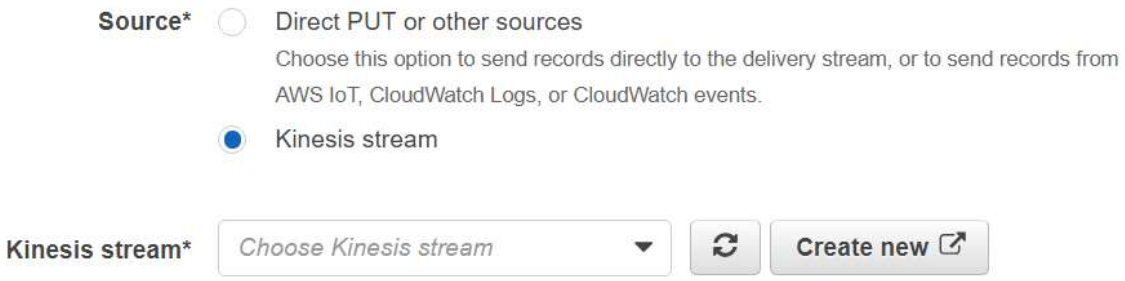
Enter a Delivery stream name: **{your alias}-traffic-message-data**

For the source there are 2 options:

1. Direct PUT or other sources
2. Kinesis stream

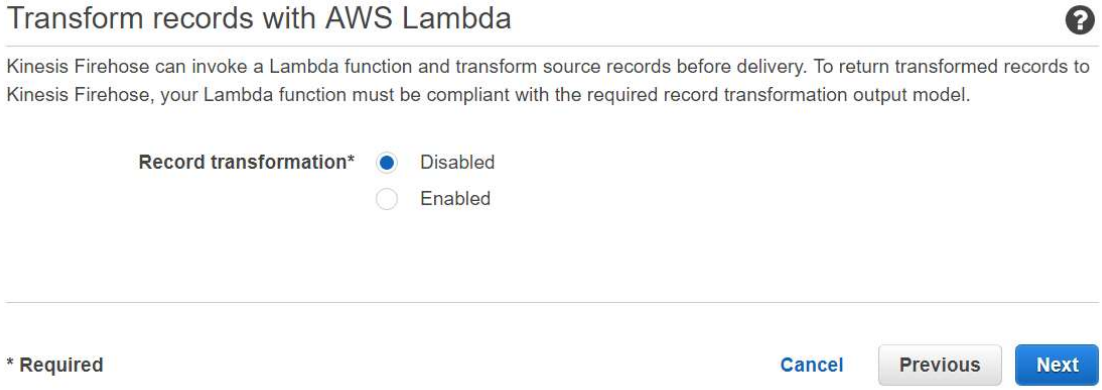
The first option allows you to directly put records on using a custom producer or from AWS IoT, Cloudwatch logs or events.
Using the second option we take an existing Kinesis stream as source.

Choose the second option.



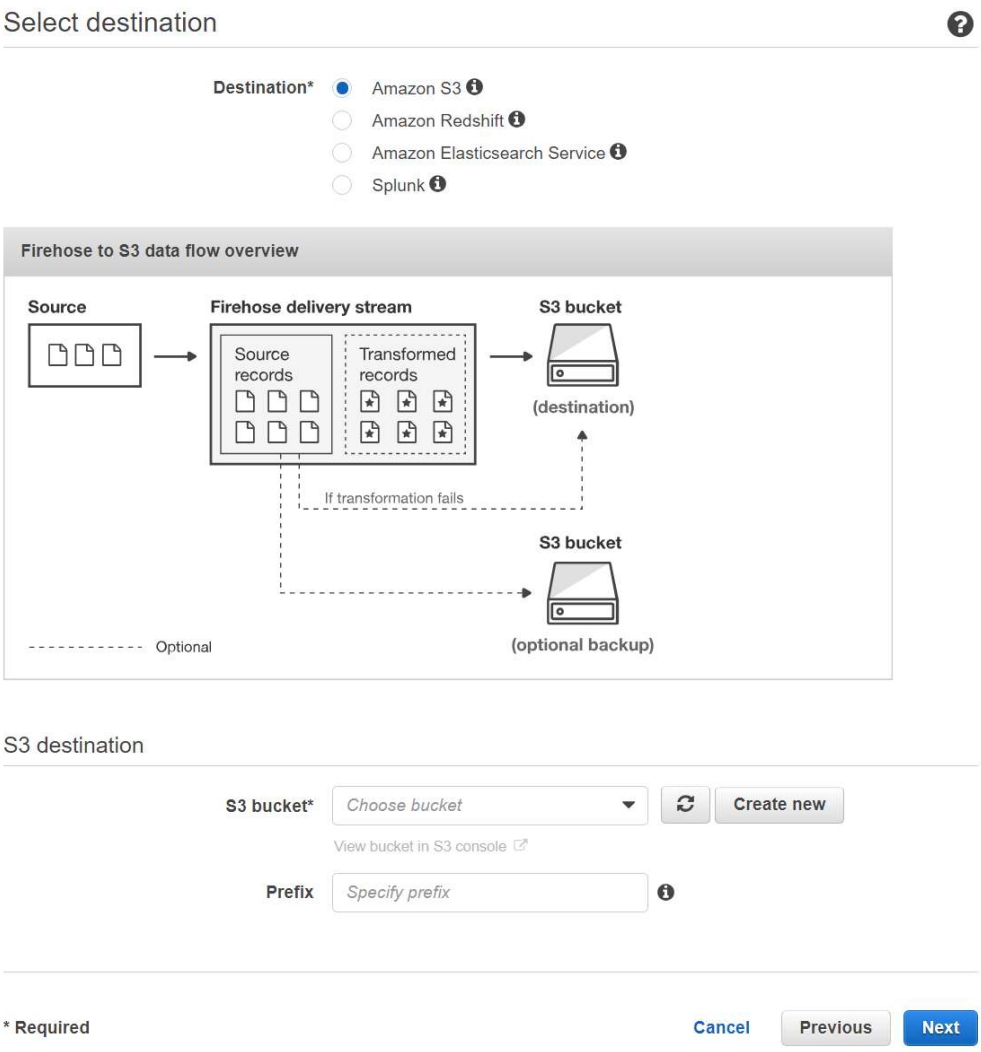
Select your stream from the dropdown box.

Press Next.



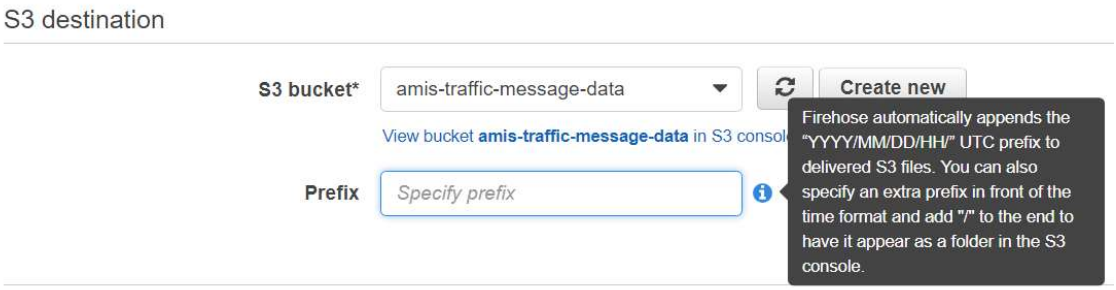
You can optionally transform the records using AWS Lambda before the records are delivered on the destination.

Select Disabled and press Next.



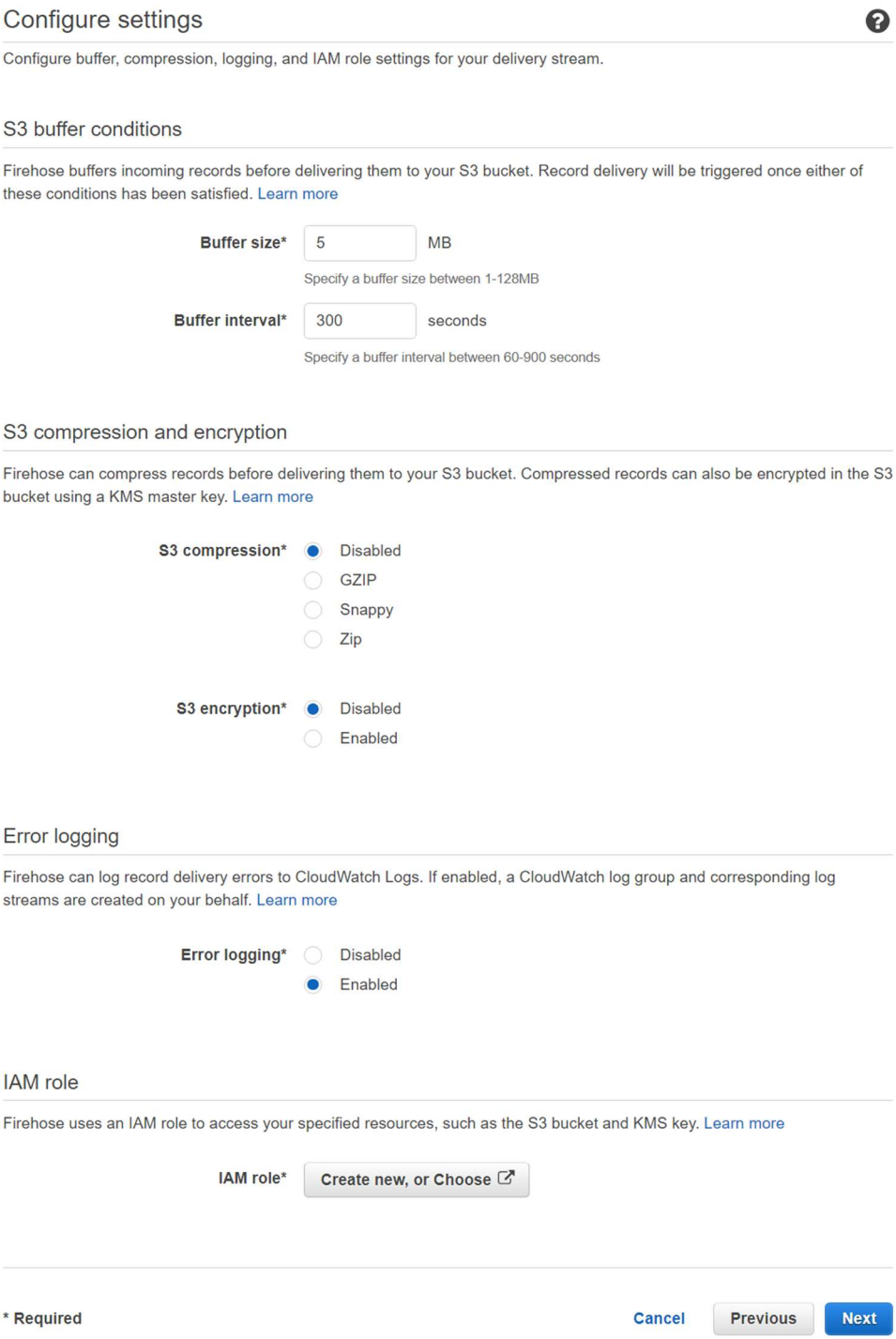
Now select the desired destination. In our case we select Amazon S3.

Select as S3 bucket: **amis-traffic-message-data**



Enter as Prefix: **{your alias}/**
Add the slash at the end to create a subfolder in the S3 bucket with your alias.

Press Next



Choose the buffer conditions. These are the conditions under which data is being offloaded.
The default settings is 5 MB or 5 minutes of data.
Just leave the default settings.
We will do no compression or encryption.

Leave Error logging enabled.

A (new) IAM role is to be created or chosen. The role is used to grant Kinesis Data Firehose access to your S3 bucket, AWS KMS key (if data encryption is enabled), and Lambda function (if data transformation is enabled).

Note the IAM role is create for you (you need IAM privileges to create the role and edit the policy).

Press the button.

Select the IAM role: **kinesis_firehose_delivery_role**

Select the Policy Name: (the only one)

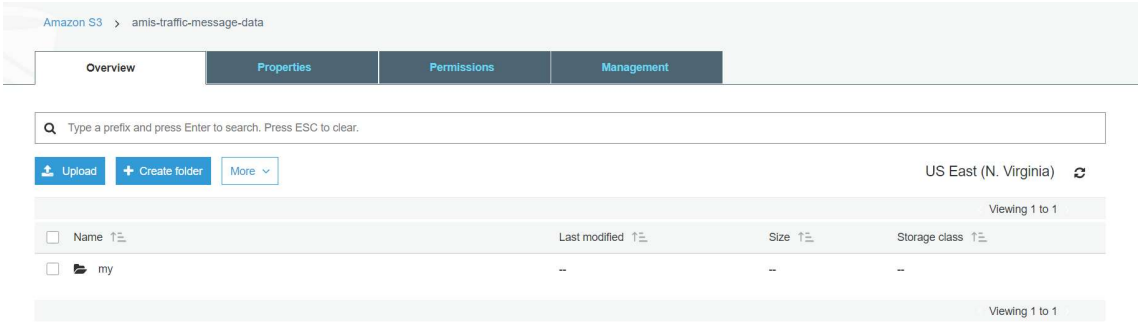
Press Allow.

Press Next.

Press Create delivery stream

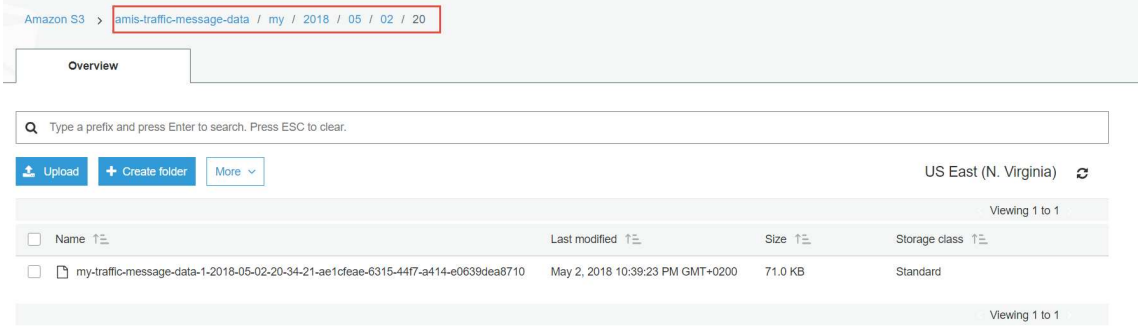
You can send some data to the stream and after at least 5 minutes, check the S3 bucket amis-traffic-message-data: https://s3.console.aws.amazon.com/s3/buckets/the-bucket/?region=us-east-1&tab=overview.

You will see a folder with your alias.



Click on the folder.

Then you will see a year folder (2018) and in that a month folder (05) and in that a day folder and then a hour folder (UTC time) and then finally your data.



Click on the file.

Press Open or Download to see the contents of the file.
In the file you will see records which have been sent to your Kinesis Stream.

Stream Consumer

It is possible to create your own custom code to read directly from a Kinesis stream. This is called a stream consumer.
In the traffic-kinesis-stream repository there is a file ‘kinesis-stream-consumer.py’. You can see this Python script for an example to dump the data from the stream to a local json file.