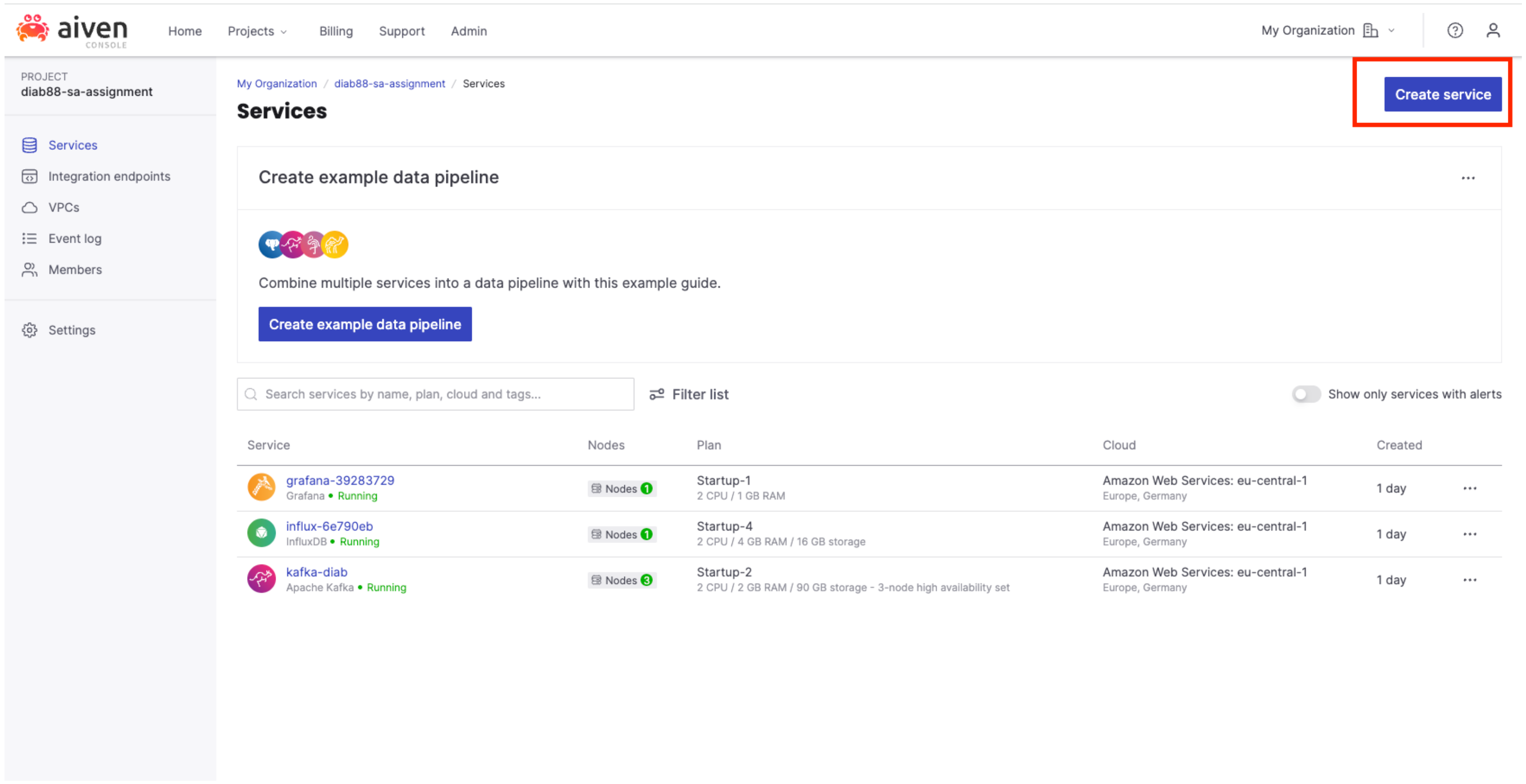
How to Create Apache Kafka Service in Aiven and Integrate It with InfluxDB and Grafana

Step 1: Login and Navigate

- Log in to your Aiven account
- From your dashboard, navigate to **"Create service"**:


aiven CONSOLE
Home
Projects
Billing
Support
Admin
My Organization
PROJECT
diab88-sa-assignment
Services
Integration endpoints
VPCs
Event log
Members
Settings
My Organization / diab88-sa-assignment / Services
Create service
Create example data pipeline
Create example data pipeline
Search services by name, plan, cloud and tags...
Filter list
Show only services with alerts

Services

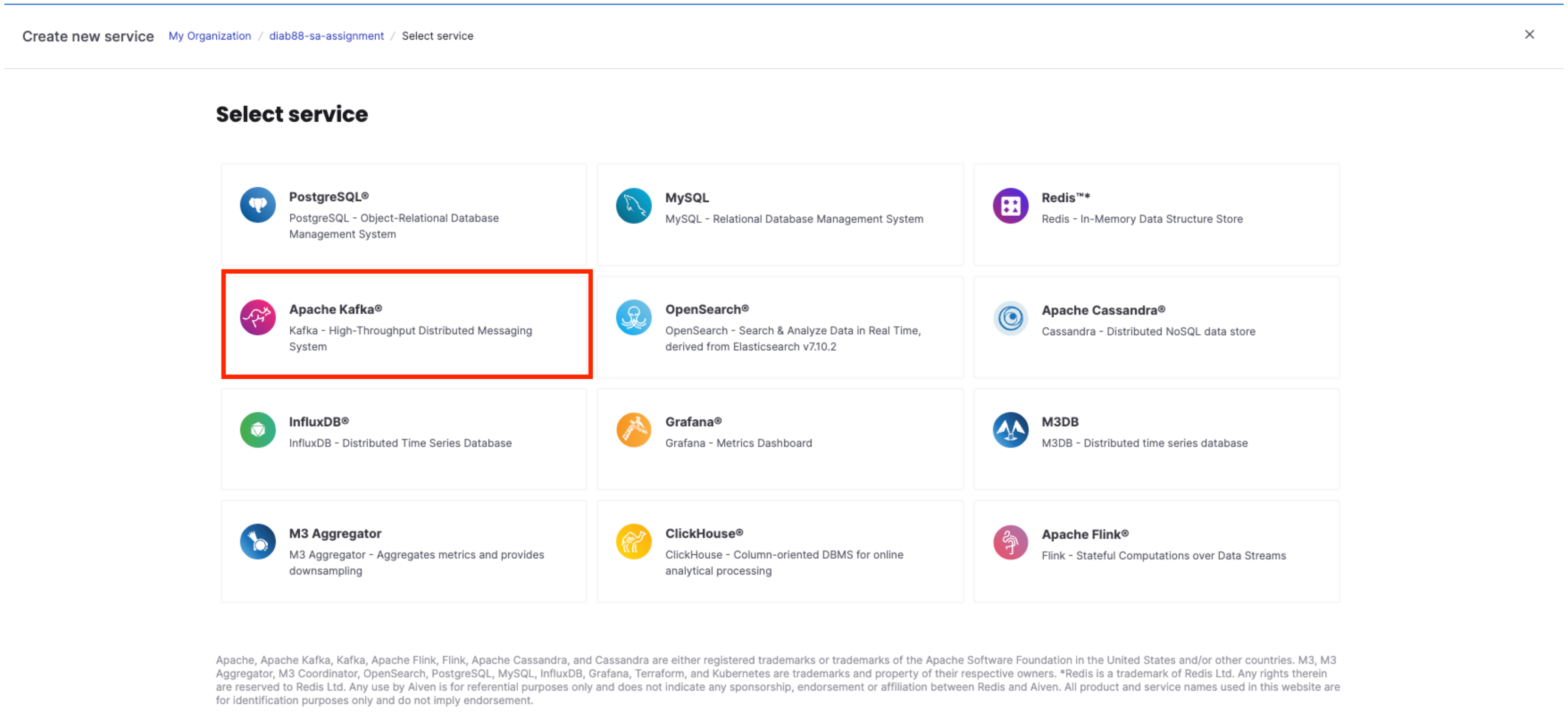
Create example data pipeline

Combine multiple services into a data pipeline with this example guide.

Service	Nodes	Plan	Cloud	Created
grafana-39283729 Grafana • Running	Nodes 1	Startup-1 2 CPU / 1 GB RAM	Amazon Web Services: eu-central-1 Europe, Germany	1 day
influx-6e790eb InfluxDB • Running	Nodes 1	Startup-4 2 CPU / 4 GB RAM / 16 GB storage	Amazon Web Services: eu-central-1 Europe, Germany	1 day
kafka-diab Apache Kafka • Running	Nodes 3	Startup-2 2 CPU / 2 GB RAM / 90 GB storage - 3-node high availability set	Amazon Web Services: eu-central-1 Europe, Germany	1 day

Step 2: Select Service Type

- On the **"Select service"** page, find and click on the **"Apache Kafka"** box:



Create new service My Organization / diab88-sa-assignment / Select service

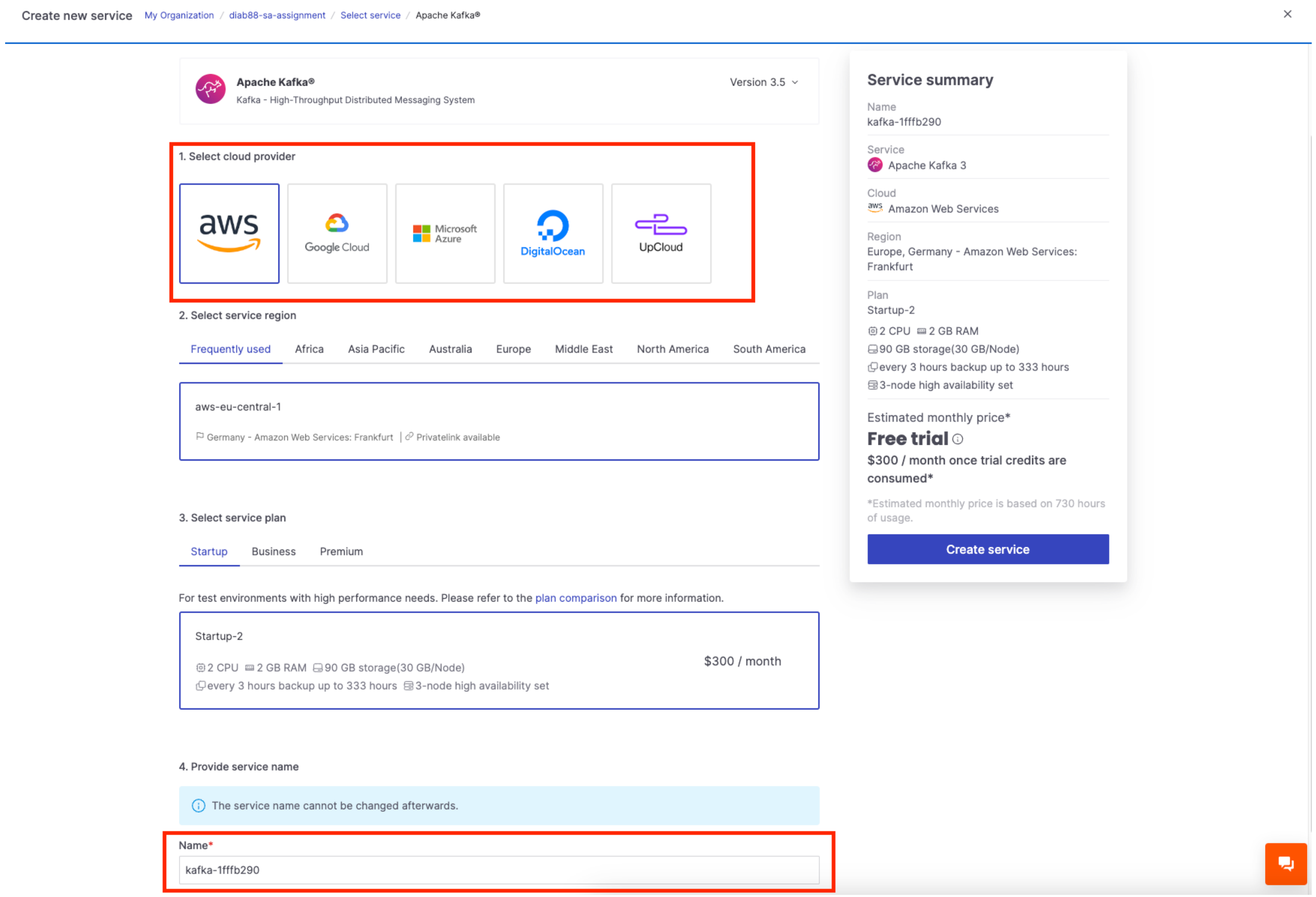
Select service

PostgreSQL® PostgreSQL - Object-Relational Database Management System	**MySQL** MySQL - Relational Database Management System	**Redis™*** Redis - In-Memory Data Structure Store
Apache Kafka® Kafka - High-Throughput Distributed Messaging System	**OpenSearch®** OpenSearch - Search & Analyze Data in Real Time, derived from Elasticsearch v7.10.2	**Apache Cassandra®** Cassandra - Distributed NoSQL data store
InfluxDB® InfluxDB - Distributed Time Series Database	**Grafana®** Grafana - Metrics Dashboard	**M3DB** M3DB - Distributed time series database
M3 Aggregator M3 Aggregator - Aggregates metrics and provides downsampling	**ClickHouse®** ClickHouse - Column-oriented DBMS for online analytical processing	**Apache Flink®** Flink - Stateful Computations over Data Streams

Apache, Apache Kafka, Kafka, Apache Flink, Flink, Apache Cassandra, and Cassandra are either registered trademarks or trademarks of the Apache Software Foundation in the United States and/or other countries. M3, M3 Aggregator, M3 Coordinator, OpenSearch, PostgreSQL, MySQL, InfluxDB, Grafana, Terraform, and Kubernetes are trademarks and property of their respective owners. *Redis is a trademark of Redis Ltd. Any rights therein are reserved to Redis Ltd. Any use by Aiven is for referential purposes only and does not indicate any sponsorship, endorsement or affiliation between Redis and Aiven. All product and service names used in this website are for identification purposes only and do not imply endorsement.

Step 3: Choose a Cloud Provider and a Service Plan

- Choose your preferred cloud provider from options like AWS, Google Cloud, Microsoft Azure. Click on your preferred provider's icon to select it
- Pick a service region
- In **"Select service plan"** section. Choose among the **"Startup", "Business",** or **"Premium"** tabs, then review the details for each plan
- Provide a service name by entering a unique name for your Apache Kafka service
- After you review your plan select **"Create Service":**


Create new service My Organization / diab88-sa-assignment / Select service / Apache Kafka®

Apache Kafka®
Kafka - High-Throughput Distributed Messaging System
Version 3.5

1. Select cloud provider

2. Select service region

Frequently used | Africa | Asia Pacific | Australia | Europe | Middle East | North America | South America

aws-eu-central-1
Germany - Amazon Web Services: Frankfurt | Privatelink available

3. Select service plan

Startup | Business | Premium

For test environments with high performance needs. Please refer to the plan comparison for more information.

Startup-2
2 CPU 2 GB RAM 90 GB storage(30 GB/Node)
every 3 hours backup up to 333 hours 3-node high availability set
$300 / month

4. Provide service name

The service name cannot be changed afterwards.

Name*
kafka-1fffb290

Service summary

Name
kafka-1fffb290

Service
Apache Kafka 3

Cloud
Amazon Web Services

Region
Europe, Germany - Amazon Web Services: Frankfurt

Plan
Startup-2
2 CPU 2 GB RAM
90 GB storage(30 GB/Node)
every 3 hours backup up to 333 hours
3-node high availability set

Estimated monthly price*
Free trial
$300 / month once trial credits are consumed*

*Estimated monthly price is based on 730 hours of usage.

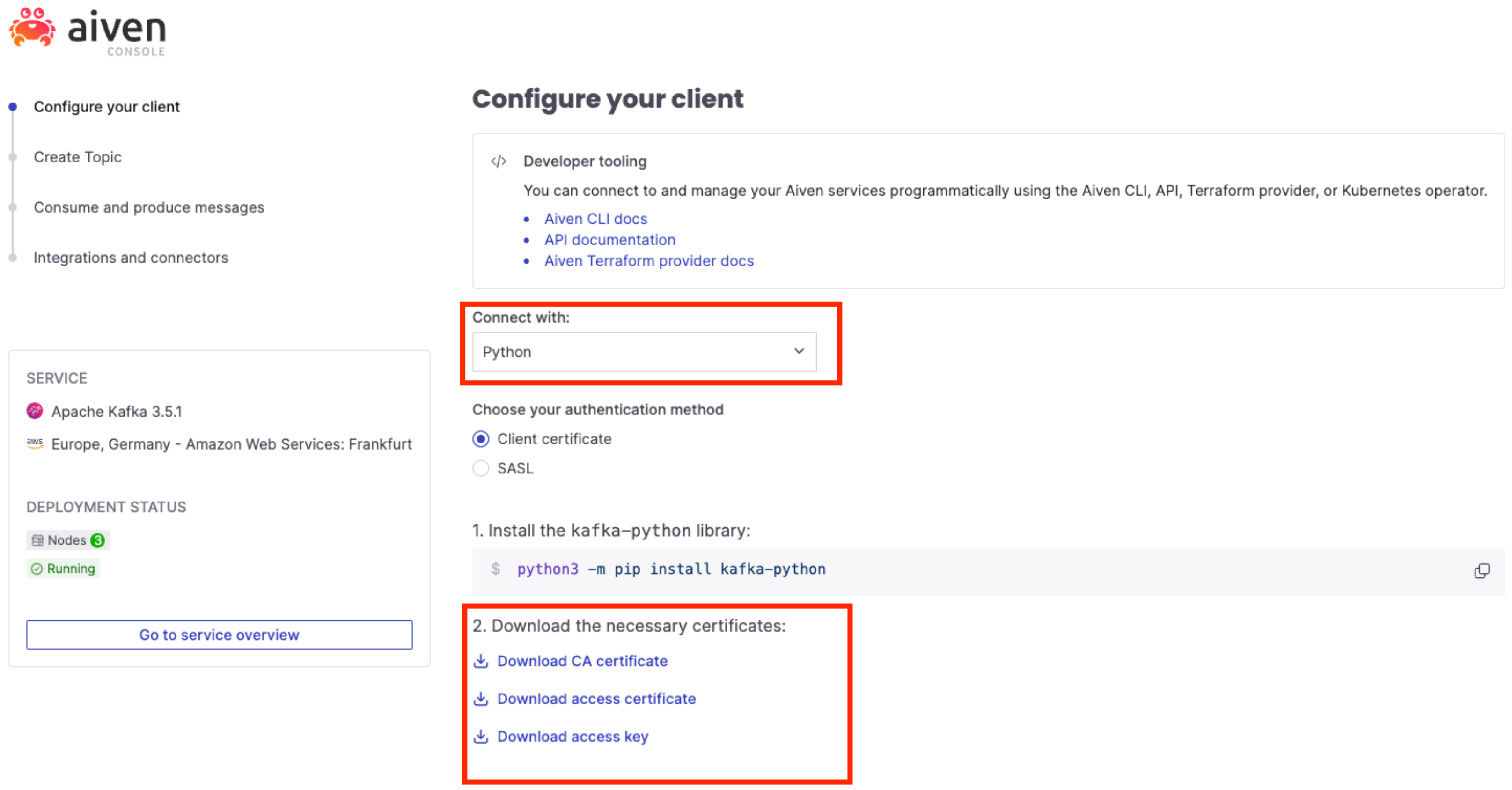
Create service

Step 4: Post-Creation

- After creating the service, it may take a few minutes for the Apache Kafka instance to be provisioned and become available
- Once the service is ready, you'll receive an email, and you can start configuring and using your Apache Kafka instance

Step 5: Configuring Kafka Client for Python

- In this example, I am choosing Python for Kafka client
- Choose **"Client certificate"** for authentication
- Download CA certificate, access certificate and access key. Ensure you keep the downloaded files secure for future use:



Configure your client

Developer tooling

You can connect to and manage your Aiven services programmatically using the Aiven CLI, API, Terraform provider, or Kubernetes operator.

- Aiven CLI docs
- API documentation
- Aiven Terraform provider docs

Connect with: Python

Choose your authentication method

[✓ Client certificate] SASL

1. Install the kafka-python library:

```
$ python3 -m pip install kafka-python
```

2. Download the necessary certificates:

- Download CA certificate
- Download access certificate
- Download access key

SERVICE

Apache Kafka 3.5.1

Europe, Germany - Amazon Web Services: Frankfurt

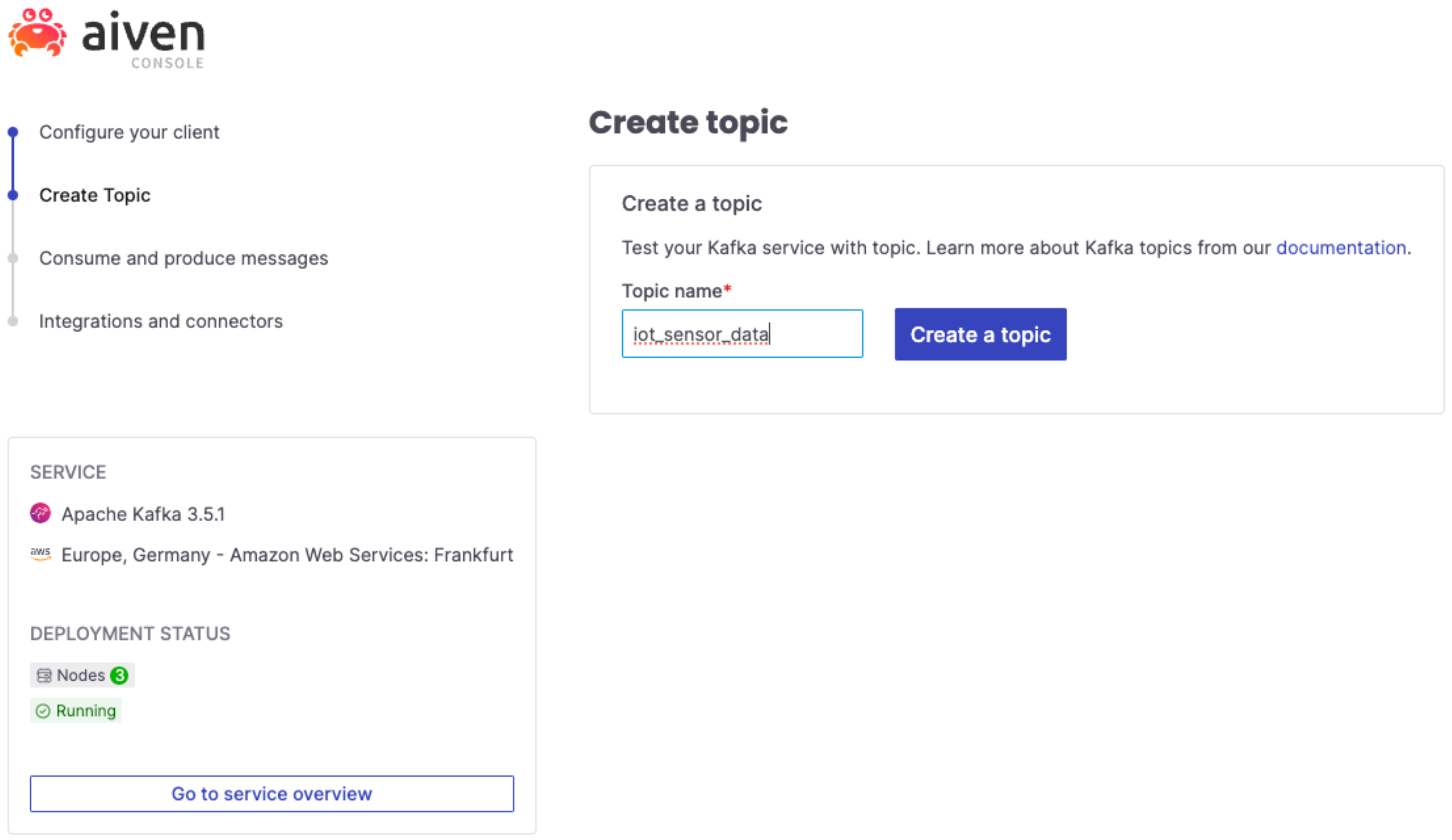
DEPLOYMENT STATUS

Nodes 3

Running

Step 6: Creating a Kafka Topic

- Click on Next and navigate to **"Create Topic"**
- Enter your desired topic name, in my case case, `iot_sensor_data`
- Click on the **"Create a topic"** button to finalize the topic creation:



Create topic

Create a topic

Test your Kafka service with topic. Learn more about Kafka topics from our documentation.

Topic name*

iot_sensor_data

SERVICE

Apache Kafka 3.5.1

Europe, Germany - Amazon Web Services: Frankfurt

DEPLOYMENT STATUS

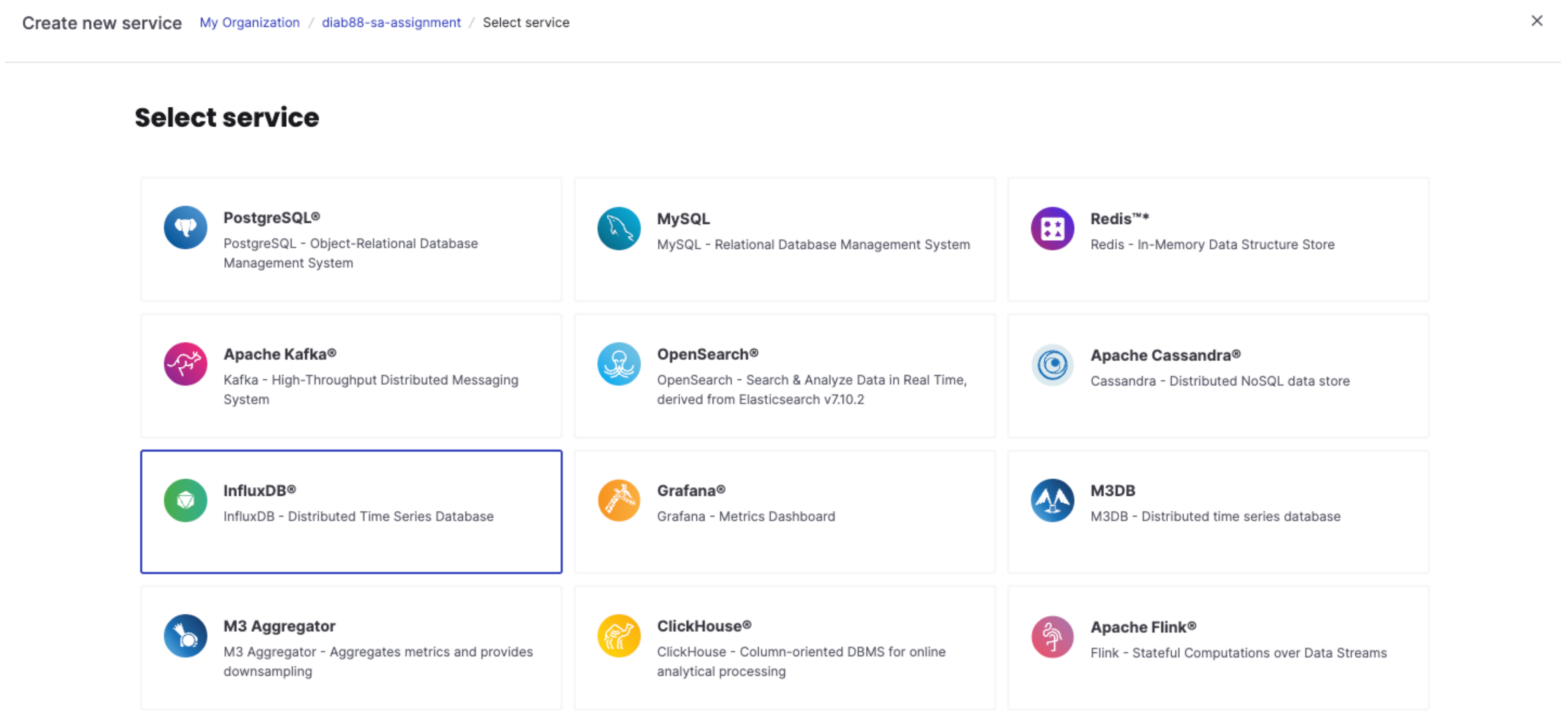
Nodes 3

Running

- Then skip the next 2 steps and click on **"Finish the setup".**

Step.7: Integrating Kafka with Influx DB

- In this step, we are going to create Influx DB first by selecting it form the service list.


Create new service My Organization / diab88-sa-assignment / Select service

Select service

PostgreSQL®
PostgreSQL - Object-Relational Database Management System

MySQL
MySQL - Relational Database Management System

Redis™*
Redis - In-Memory Data Structure Store

Apache Kafka®
Kafka - High-Throughput Distributed Messaging System

OpenSearch®
OpenSearch - Search & Analyze Data in Real Time, derived from Elasticsearch v7.10.2

Apache Cassandra®
Cassandra - Distributed NoSQL data store

InfluxDB®
InfluxDB - Distributed Time Series Database

Grafana®
Grafana - Metrics Dashboard

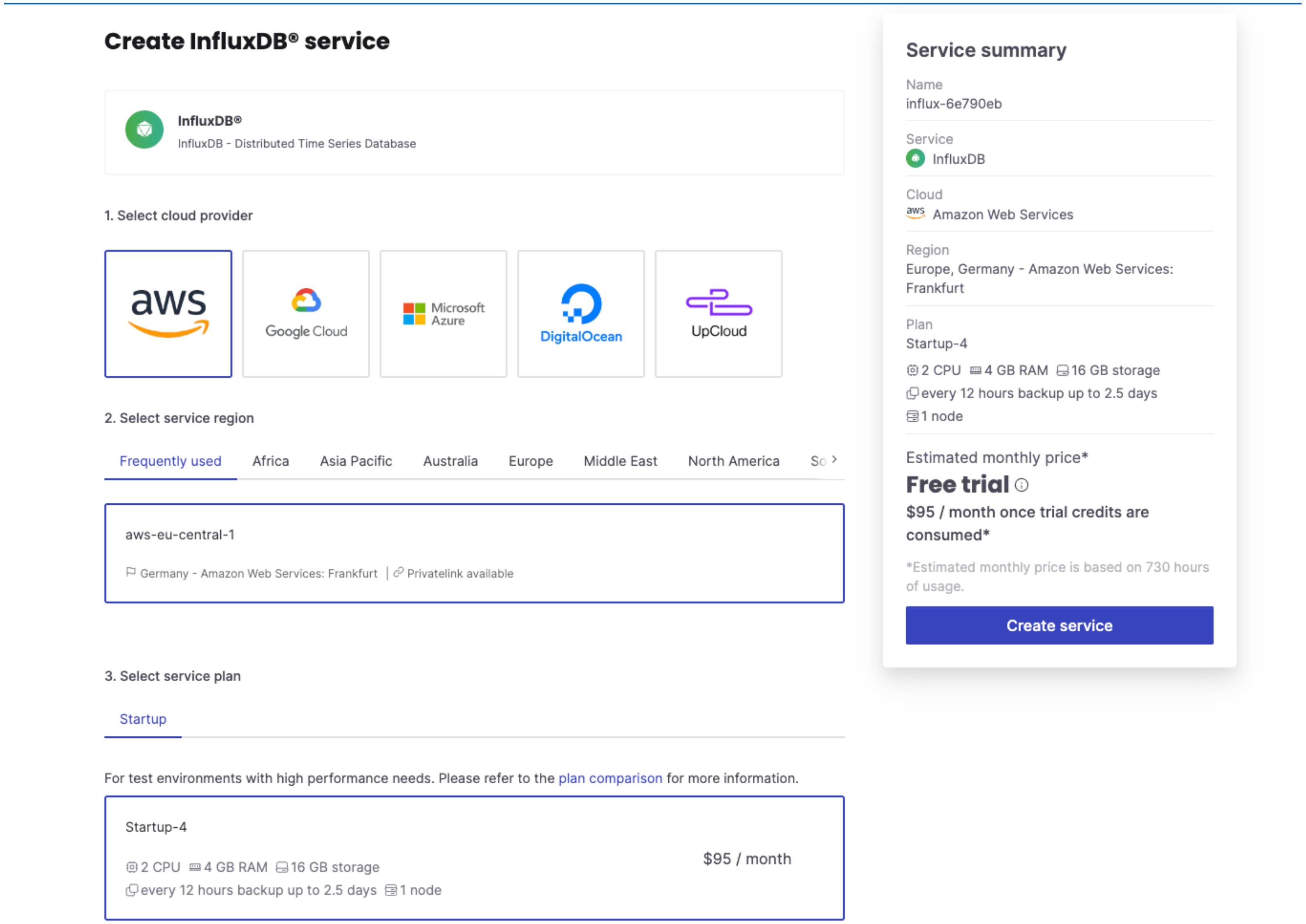
M3DB
M3DB - Distributed time series database

M3 Aggregator
M3 Aggregator - Aggregates metrics and provides downsampling

ClickHouse®
ClickHouse - Column-oriented DBMS for online analytical processing

Apache Flink®
Flink - Stateful Computations over Data Streams

- Next step is to choose the cloud provider, region, and service plan for Influx DB, then click on **“Create Service”** after choosing a proper name:



Create InfluxDB® service

InfluxDB®
InfluxDB - Distributed Time Series Database

1. Select cloud provider

2. Select service region

aws-eu-central-1
Germany - Amazon Web Services: Frankfurt | Privatelink available

3. Select service plan

For test environments with high performance needs. Please refer to the plan comparison for more information.

Startup-4
2 CPU 4 GB RAM 16 GB storage
every 12 hours backup up to 2.5 days 1 node
$95 / month

Service summary

Name
influx-6e790eb

Service
InfluxDB

Cloud
Amazon Web Services

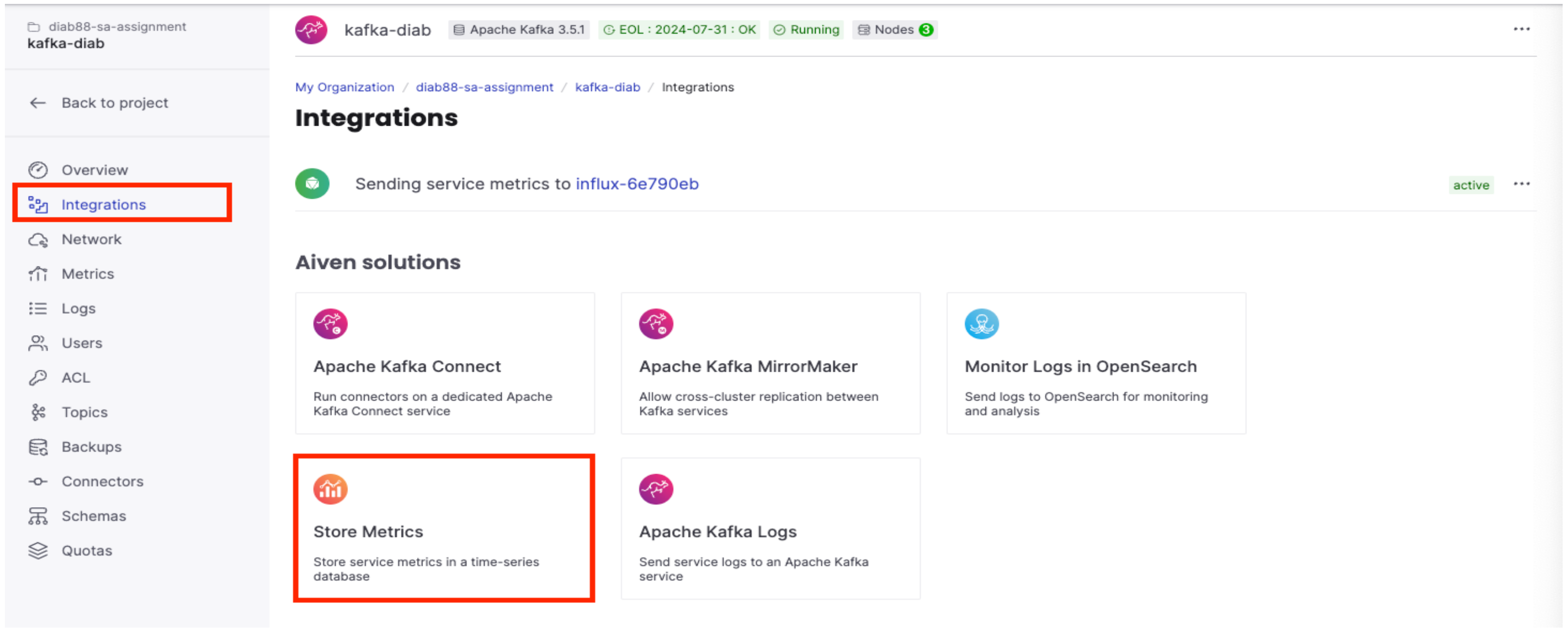
Region
Europe, Germany - Amazon Web Services: Frankfurt

Plan
Startup-4
2 CPU 4 GB RAM 16 GB storage
every 12 hours backup up to 2.5 days
1 node

Estimated monthly price*
Free trial
$95 / month once trial credits are consumed*

*Estimated monthly price is based on 730 hours of usage.

- It may take a few minutes for instance to be provisioned. Once the service is ready, you'll receive an email
- Now by going back to the Kafka service, on the side bar click on **“Integration”,** choose **“Store Metrics”:**


diab88-sa-assignment
kafka-diab

kafka-diab · Apache Kafka 3.5.1 · EOL : 2024-07-31 : OK · Running · Nodes 3

My Organization / diab88-sa-assignment / kafka-diab / Integrations

Integrations

Sending service metrics to influx-6e790eb · active

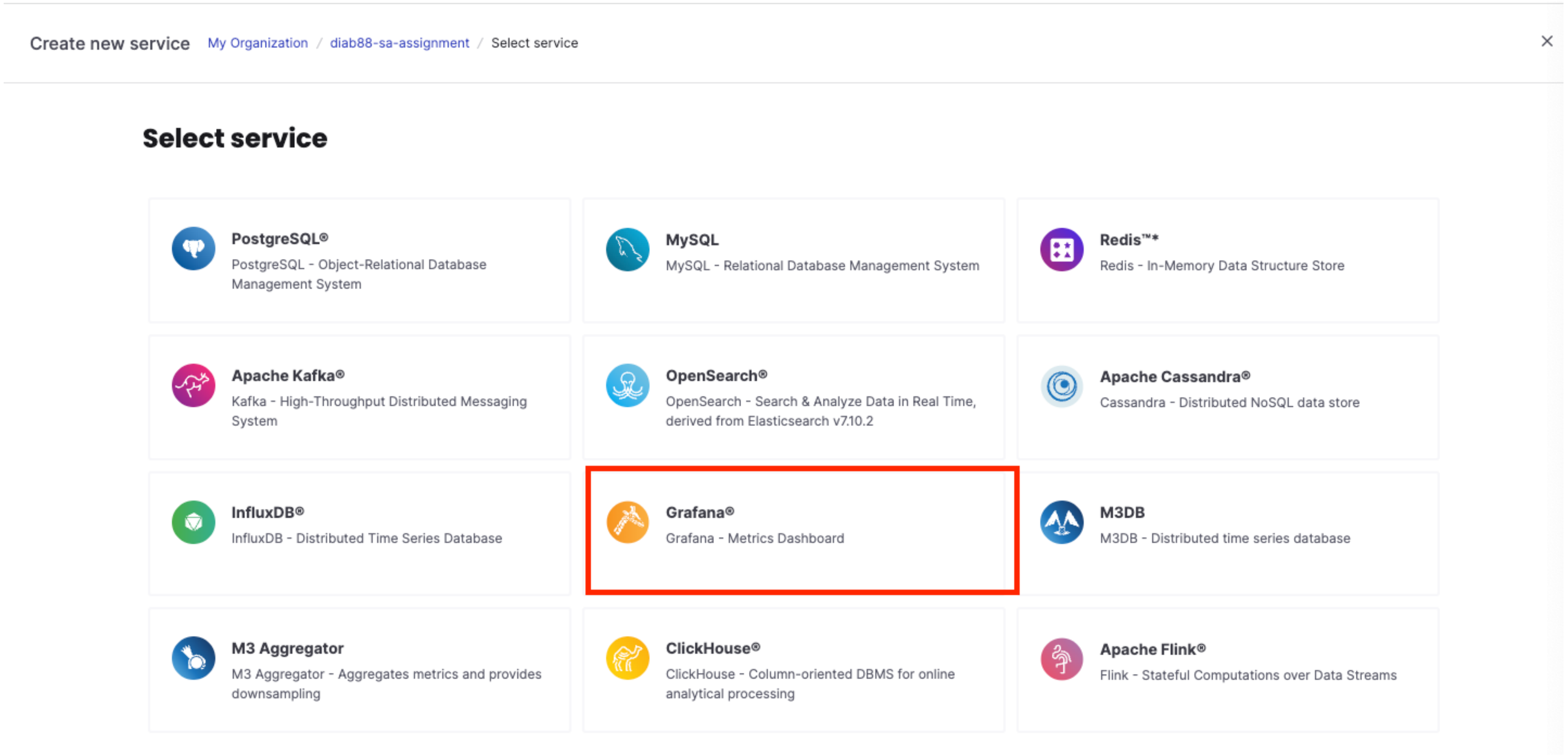
Aiven solutions

- Apache Kafka Connect — Run connectors on a dedicated Apache Kafka Connect service
- Apache Kafka MirrorMaker — Allow cross-cluster replication between Kafka services
- Monitor Logs in OpenSearch — Send logs to OpenSearch for monitoring and analysis
- Store Metrics — Store service metrics in a time-series database
- Apache Kafka Logs — Send service logs to an Apache Kafka service

- Then choose the Influx DB which you have already created to be integrated with Kafka and the metrics will be sent to it.

Step 8: Creating Grafana for Monitoring and Metrics Observability

- In this step, we are going to create Grafana by selecting it from service list
- Next step is to choose the cloud provider, region, and service plan for Grafana service, then click on **"Create Service"** after choosing a proper name.



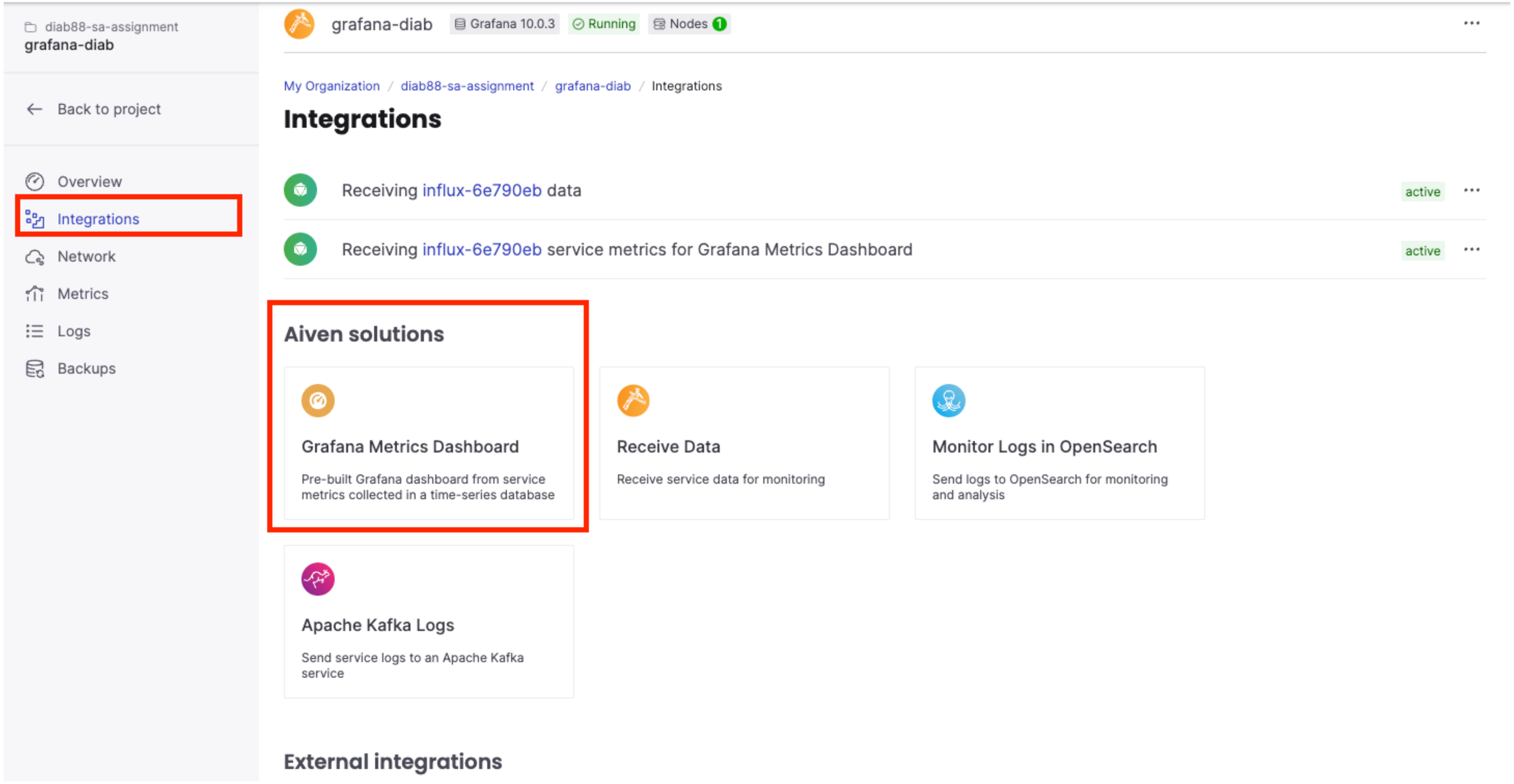
Create new service My Organization / diab88-sa-assignment / Select service

Select service

PostgreSQL® PostgreSQL - Object-Relational Database Management System	**MySQL** MySQL - Relational Database Management System	**Redis™*** Redis - In-Memory Data Structure Store
Apache Kafka® Kafka - High-Throughput Distributed Messaging System	**OpenSearch®** OpenSearch - Search & Analyze Data in Real Time, derived from Elasticsearch v7.10.2	**Apache Cassandra®** Cassandra - Distributed NoSQL data store
InfluxDB® InfluxDB - Distributed Time Series Database	**Grafana®** Grafana - Metrics Dashboard	**M3DB** M3DB - Distributed time series database
M3 Aggregator M3 Aggregator - Aggregates metrics and provides downsampling	**ClickHouse®** ClickHouse - Column-oriented DBMS for online analytical processing	**Apache Flink®** Flink - Stateful Computations over Data Streams

- It may take a few minutes for instance to be provisioned. Once the service is ready, you'll receive an email.

- From Grafana portal on the side bar, choose **"Integration"** then click on **"Grafana Metrics Dashboard"** and select to integrate with **"InfluxDB"** which you have just created:



diab88-sa-assignment
grafana-diab

Back to project

Overview
Integrations
Network
Metrics
Logs
Backups

grafana-diab   Grafana 10.0.3   Running   Nodes 1

My Organization / diab88-sa-assignment / grafana-diab / Integrations

Integrations

Receiving influx-6e790eb data   active

Receiving influx-6e790eb service metrics for Grafana Metrics Dashboard   active

Aiven solutions

Grafana Metrics Dashboard
Pre-built Grafana dashboard from service metrics collected in a time-series database

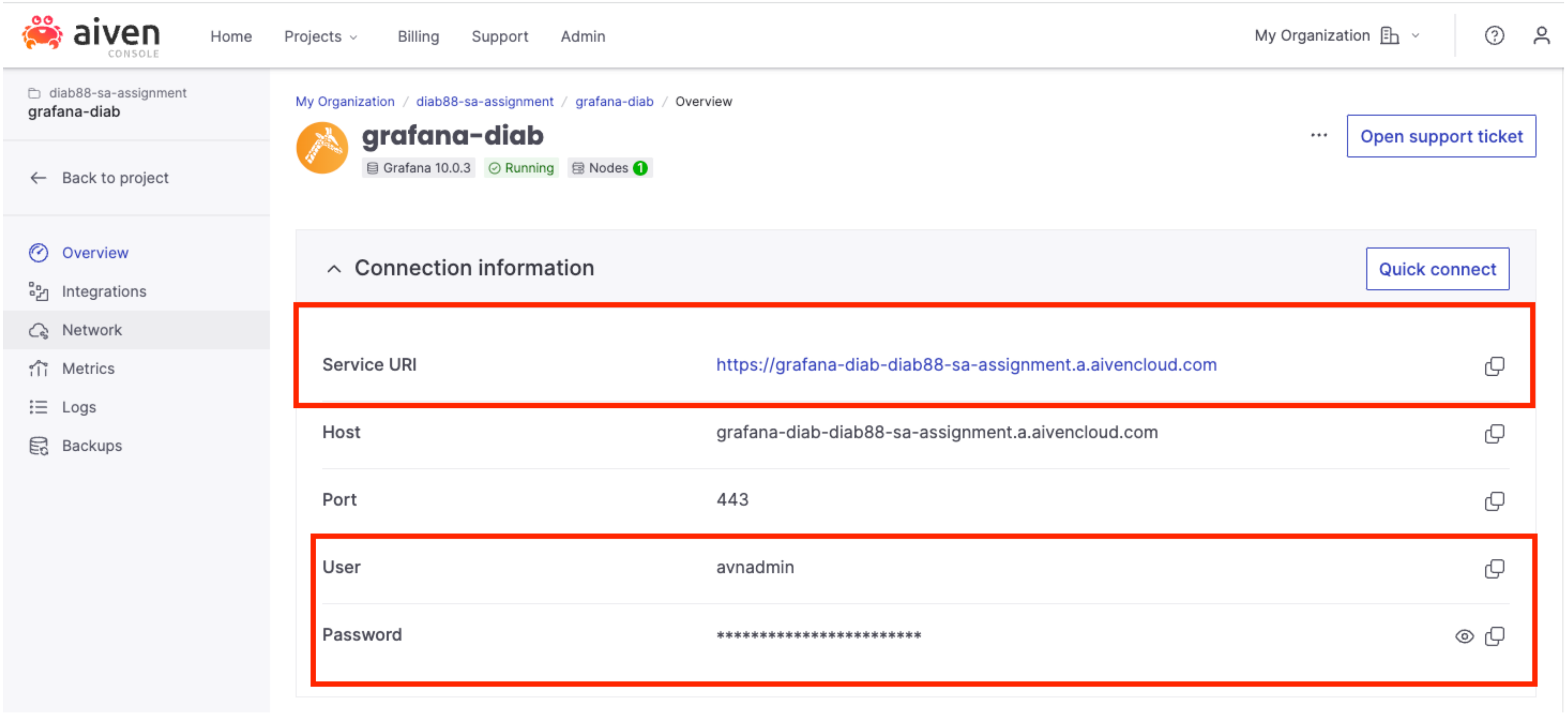
Receive Data
Receive service data for monitoring

Monitor Logs in OpenSearch
Send logs to OpenSearch for monitoring and analysis

Apache Kafka Logs
Send service logs to an Apache Kafka service

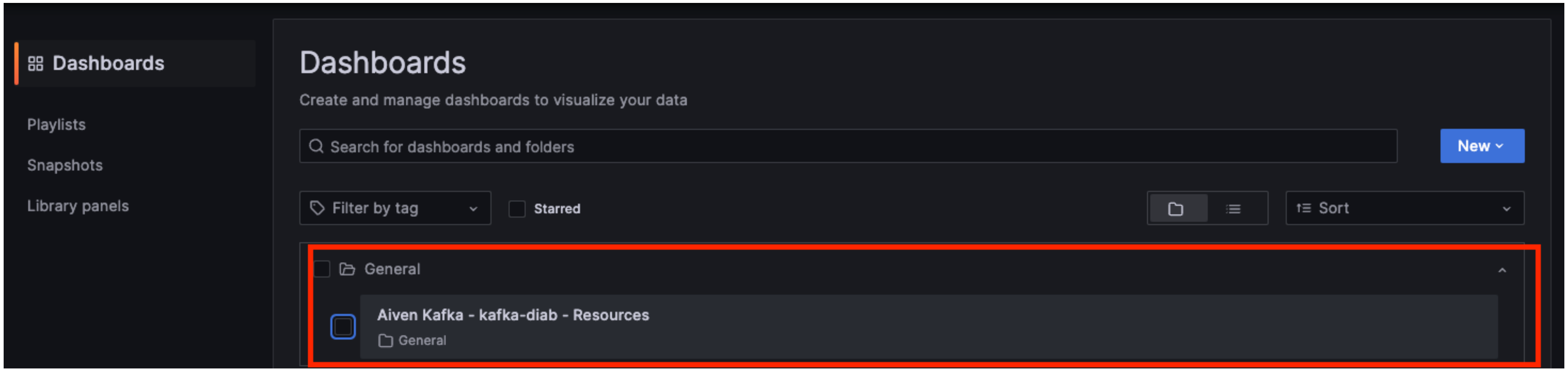
External integrations

- In the Overview section, you can find the URl, User and password to connect to the Grafana dashboard:



Service URI	https://grafana-diab-diab88-sa-assignment.a.aivencloud.com
Host	grafana-diab-diab88-sa-assignment.a.aivencloud.com
Port	443
User	avnadmin
Password	************************

- After logging into Grafana, navigate to sidebar, click on **“Dashboard”,** you will find a pre-built Grafana dashboard representing service metrics collected in the InluxDb:



Dashboards

Create and manage dashboards to visualize your data

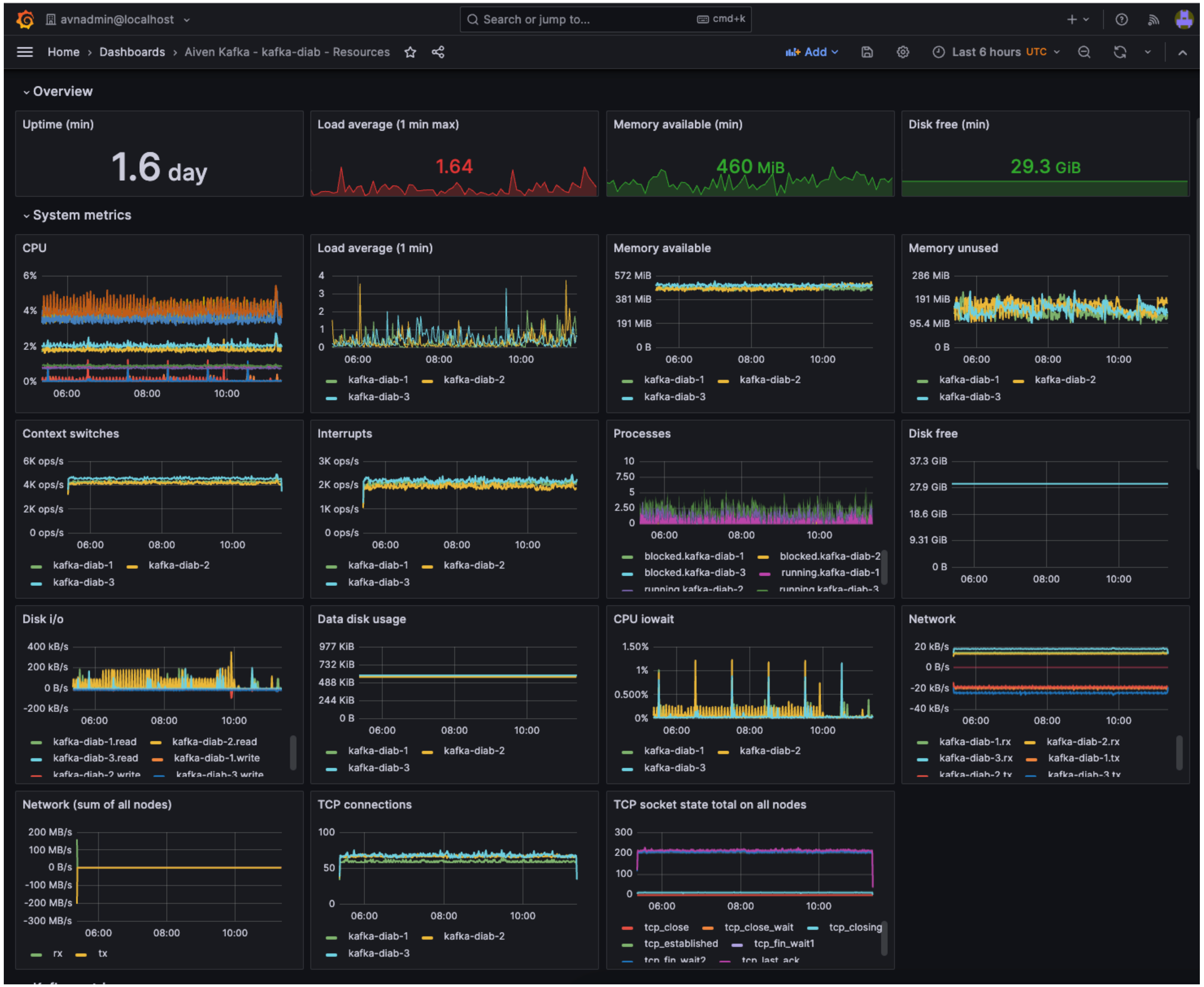
Search for dashboards and folders

General

Aiven Kafka - kafka-diab - Resources

General

- Click on it, and then it will open a wonderful dashboard with all the required metrics:



Step 9: Send IOT sensor data to the Kafka topic

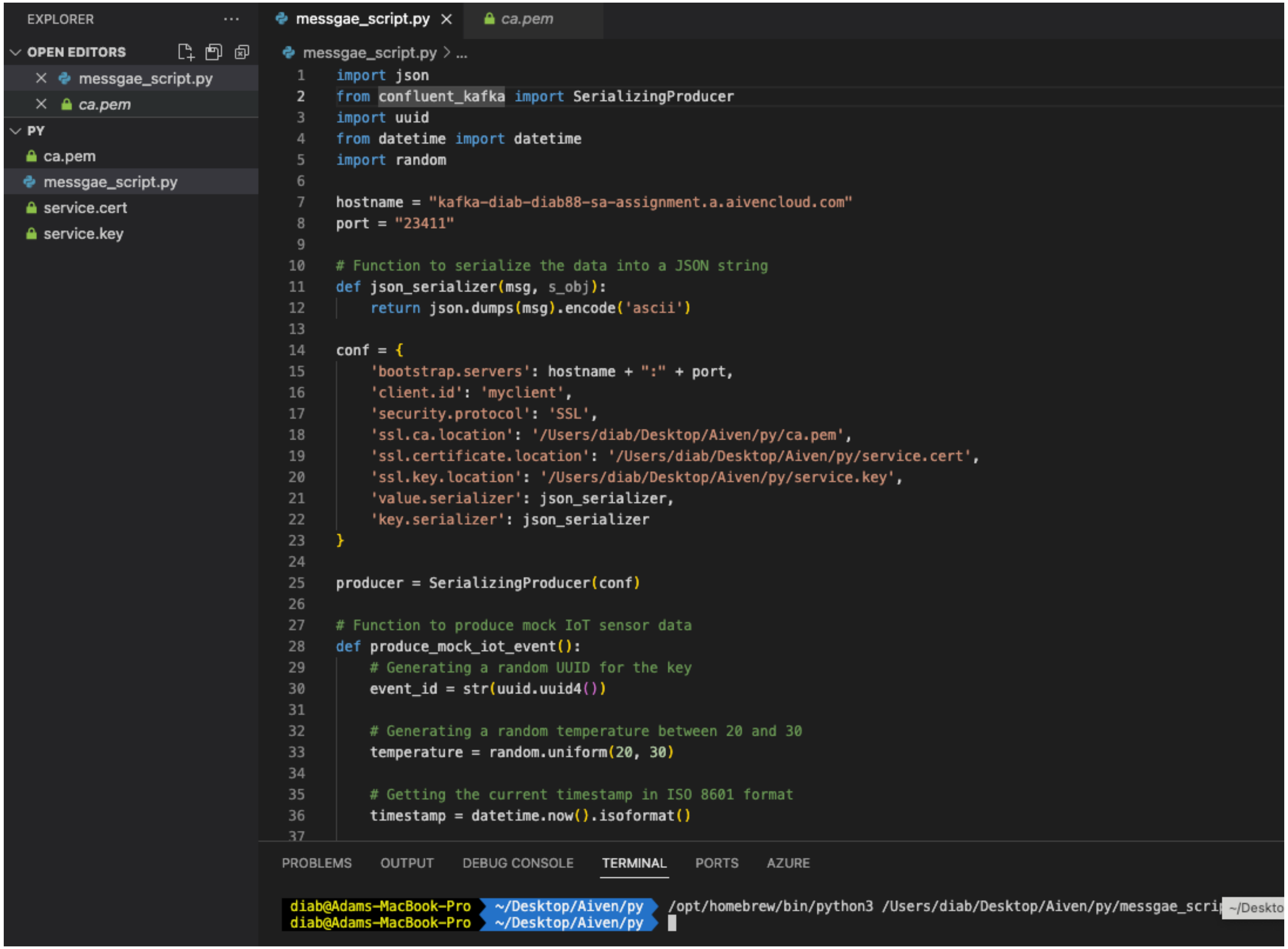
In this step, we will be using a Python script hosted in this public repository here. It simulates the production of IoT sensor data events and sends them to a Kafka topic.

It generates mock sensor data such as temperature, timestamp, and location, and send this data to the **iot_sensor_data** topic which has been created in Kafka configuration **Step 6**.

To run the code:

- Prepare Python environment: Python 3.11.X
- Install **confluent_kafka** Python library: **pip install confluent-kafka**
- Ensure you have the necessary SSL certificates which you have downloaded in Step 5 at the specified locations on your machine or, if you have your SSL certificates at different locations, update the paths in the script
- Run the script.

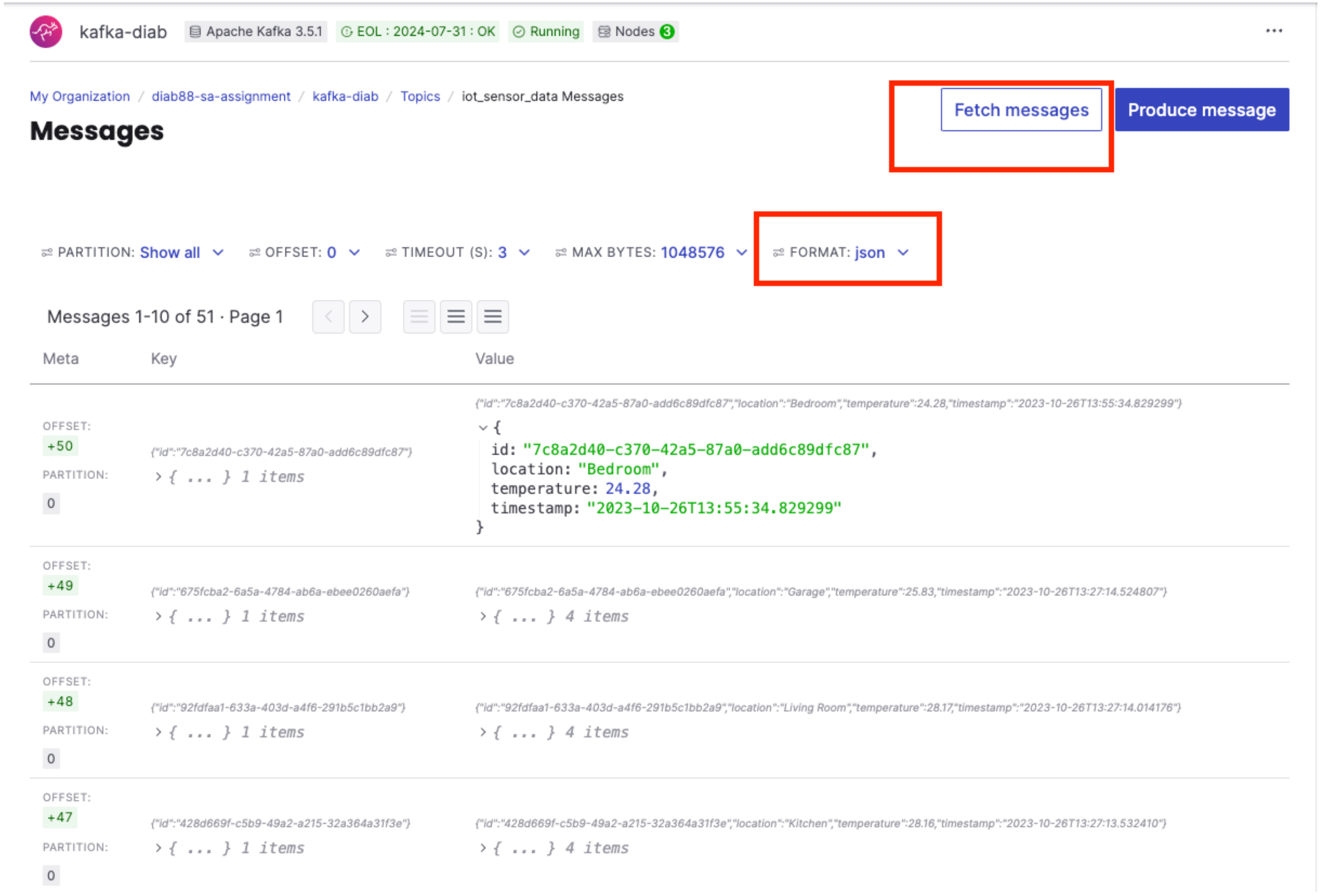
Now in this showcase I am running the code on my local machine:



```python
import json
from confluent_kafka import SerializingProducer
import uuid
from datetime import datetime
import random

hostname = "kafka-diab-diab88-sa-assignment.a.aivencloud.com"
port = "23411"

# Function to serialize the data into a JSON string
def json_serializer(msg, s_obj):
    return json.dumps(msg).encode('ascii')

conf = {
    'bootstrap.servers': hostname + ":" + port,
    'client.id': 'myclient',
    'security.protocol': 'SSL',
    'ssl.ca.location': '/Users/diab/Desktop/Aiven/py/ca.pem',
    'ssl.certificate.location': '/Users/diab/Desktop/Aiven/py/service.cert',
    'ssl.key.location': '/Users/diab/Desktop/Aiven/py/service.key',
    'value.serializer': json_serializer,
    'key.serializer': json_serializer
}

producer = SerializingProducer(conf)

# Function to produce mock IoT sensor data
def produce_mock_iot_event():
    # Generating a random UUID for the key
    event_id = str(uuid.uuid4())

    # Generating a random temperature between 20 and 30
    temperature = random.uniform(20, 30)

    # Getting the current timestamp in ISO 8601 format
    timestamp = datetime.now().isoformat()

```

```
diab@Adams-MacBook-Pro  ~/Desktop/Aiven/py  /opt/homebrew/bin/python3 /Users/diab/Desktop/Aiven/py/messgae_scri ~/Deskto
diab@Adams-MacBook-Pro  ~/Desktop/Aiven/py  
```

- To check the message in the Kafka topic, go again to Kafka service. Then on the right console click on **“Topics”.** You will find the topic you have created. In my case it is **Iot_sensor_data,** then click on **“Messages”**.
- Click on “**Fetch messages”** after choosing a Json format, then all the sent messages will appear in a Json format:



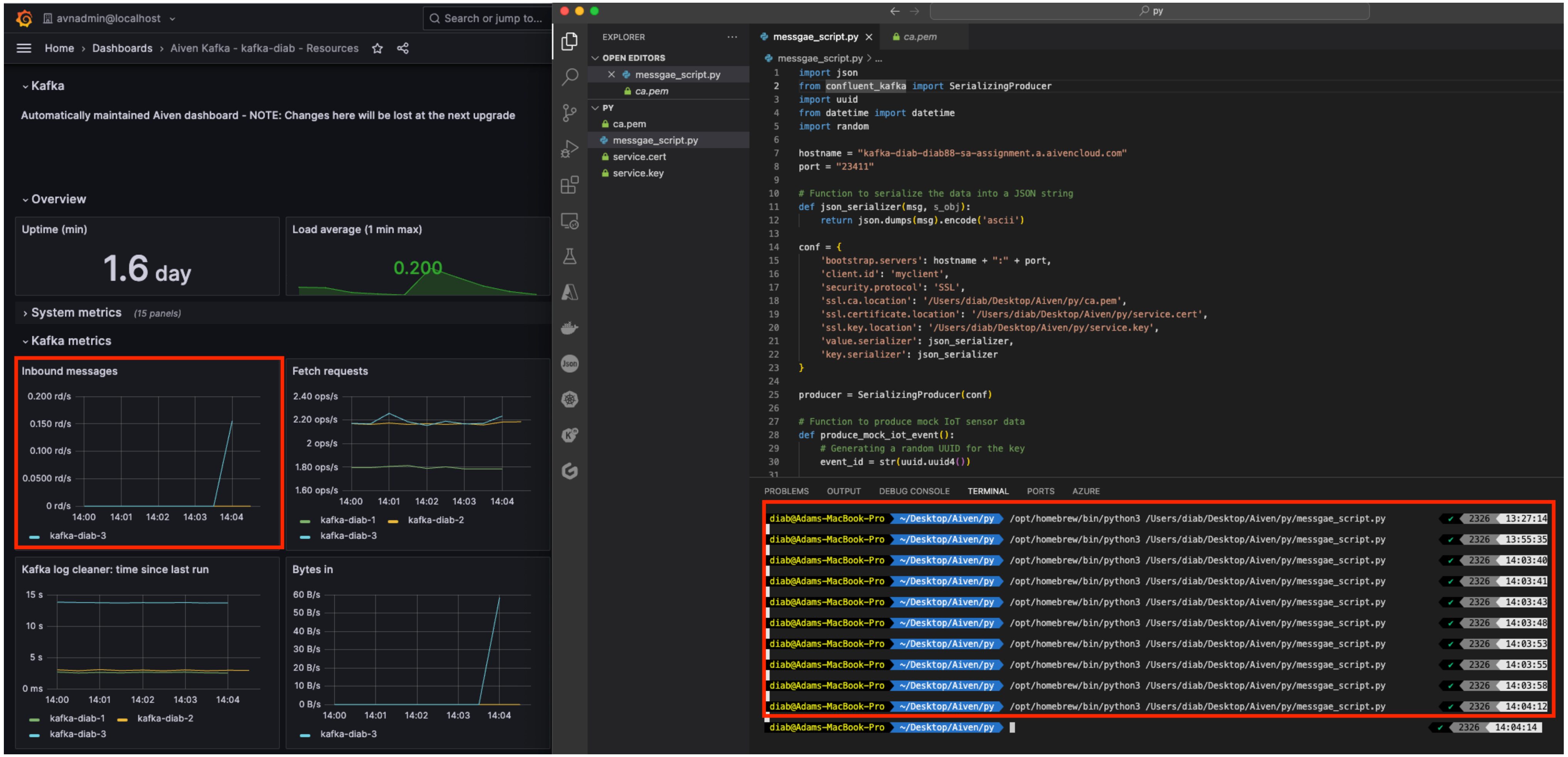
Messages

PARTITION: Show all OFFSET: 0 TIMEOUT (S): 3 MAX BYTES: 1048576 FORMAT: json

Messages 1-10 of 51 · Page 1

```
Meta        Key                                              Value
OFFSET:                                                      {"id":"7c8a2d40-c370-42a5-87a0-add6c89dfc87","location":"Bedroom","temperature":24.28,"timestamp":"2023-10-26T13:55:34.829299"}
+50         {"id":"7c8a2d40-c370-42a5-87a0-add6c89dfc87"}     ⌄{
PARTITION:  › { ... } 1 items                                  id: "7c8a2d40-c370-42a5-87a0-add6c89dfc87",
0                                                              location: "Bedroom",
                                                               temperature: 24.28,
                                                               timestamp: "2023-10-26T13:55:34.829299"
                                                             }
OFFSET:
+49         {"id":"675fcba2-6a5a-4784-ab6a-ebee0260aefa"}     {"id":"675fcba2-6a5a-4784-ab6a-ebee0260aefa","location":"Garage","temperature":25.83,"timestamp":"2023-10-26T13:27:14.524807"}
PARTITION:  › { ... } 1 items                                › { ... } 4 items
0
OFFSET:
+48         {"id":"92fdfaa1-633a-403d-a4f6-291b5c1bb2a9"}     {"id":"92fdfaa1-633a-403d-a4f6-291b5c1bb2a9","location":"Living Room","temperature":28.17,"timestamp":"2023-10-26T13:27:14.014176"}
PARTITION:  › { ... } 1 items                                › { ... } 4 items
0
OFFSET:
+47         {"id":"428d669f-c5b9-49a2-a215-32a364a31f3e"}     {"id":"428d669f-c5b9-49a2-a215-32a364a31f3e","location":"Kitchen","temperature":28.16,"timestamp":"2023-10-26T13:27:13.532410"}
PARTITION:  › { ... } 1 items                                › { ... } 4 items
0
```

- By sending more messages to the Kafka topic, we can see a spike in the Kafka inbound message on Grafana dashboard:



```python
import json
from confluent_kafka import SerializingProducer
import uuid
from datetime import datetime
import random

hostname = "kafka-diab-diab88-sa-assignment.a.aivencloud.com"
port = "23411"

# Function to serialize the data into a JSON string
def json_serializer(msg, s_obj):
    return json.dumps(msg).encode('ascii')

conf = {
    'bootstrap.servers': hostname + ":" + port,
    'client.id': 'myclient',
    'security.protocol': 'SSL',
    'ssl.ca.location': '/Users/diab/Desktop/Aiven/py/ca.pem',
    'ssl.certificate.location': '/Users/diab/Desktop/Aiven/py/service.cert',
    'ssl.key.location': '/Users/diab/Desktop/Aiven/py/service.key',
    'value.serializer': json_serializer,
    'key.serializer': json_serializer
}

producer = SerializingProducer(conf)

# Function to produce mock IoT sensor data
def produce_mock_iot_event():
    # Generating a random UUID for the key
    event_id = str(uuid.uuid4())
```

```
diab@Adams-MacBook-Pro  ~/Desktop/Aiven/py  /opt/homebrew/bin/python3 /Users/diab/Desktop/Aiven/py/messgae_script.py     ✓  2326  13:27:14
diab@Adams-MacBook-Pro  ~/Desktop/Aiven/py  /opt/homebrew/bin/python3 /Users/diab/Desktop/Aiven/py/messgae_script.py     ✓  2326  13:55:35
diab@Adams-MacBook-Pro  ~/Desktop/Aiven/py  /opt/homebrew/bin/python3 /Users/diab/Desktop/Aiven/py/messgae_script.py     ✓  2326  14:03:40
diab@Adams-MacBook-Pro  ~/Desktop/Aiven/py  /opt/homebrew/bin/python3 /Users/diab/Desktop/Aiven/py/messgae_script.py     ✓  2326  14:03:41
diab@Adams-MacBook-Pro  ~/Desktop/Aiven/py  /opt/homebrew/bin/python3 /Users/diab/Desktop/Aiven/py/messgae_script.py     ✓  2326  14:03:43
diab@Adams-MacBook-Pro  ~/Desktop/Aiven/py  /opt/homebrew/bin/python3 /Users/diab/Desktop/Aiven/py/messgae_script.py     ✓  2326  14:03:48
diab@Adams-MacBook-Pro  ~/Desktop/Aiven/py  /opt/homebrew/bin/python3 /Users/diab/Desktop/Aiven/py/messgae_script.py     ✓  2326  14:03:53
diab@Adams-MacBook-Pro  ~/Desktop/Aiven/py  /opt/homebrew/bin/python3 /Users/diab/Desktop/Aiven/py/messgae_script.py     ✓  2326  14:03:55
diab@Adams-MacBook-Pro  ~/Desktop/Aiven/py  /opt/homebrew/bin/python3 /Users/diab/Desktop/Aiven/py/messgae_script.py     ✓  2326  14:03:58
diab@Adams-MacBook-Pro  ~/Desktop/Aiven/py  /opt/homebrew/bin/python3 /Users/diab/Desktop/Aiven/py/messgae_script.py     ✓  2326  14:04:12
diab@Adams-MacBook-Pro  ~/Desktop/Aiven/py                                                                              ✓  2326  14:04:14
```