Aiven-Kafka-Python-Quickstart

## A simple tutorial to get started using Kafka services managed by Aiven

Aiven (pronounced ai-vn) makes it convenient to quickly experiment with many types of data stores. Today we're going to be taking a look at how to quickly setup an Aiven account, create a Kafka service with observability and monitoring through Grafana, and demonstrate it's use by emulating an IoT device that publishes simple weather data to Kafka.

## Prerequisites

This tutorial assumes you have Python installed and correctly configured on your system. You will need to ensure that you have the Kafka-Python and timezone packages installed. This can be accomplished using pip.

pip install kafka-python

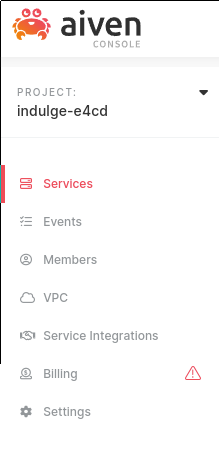
pip install timezone

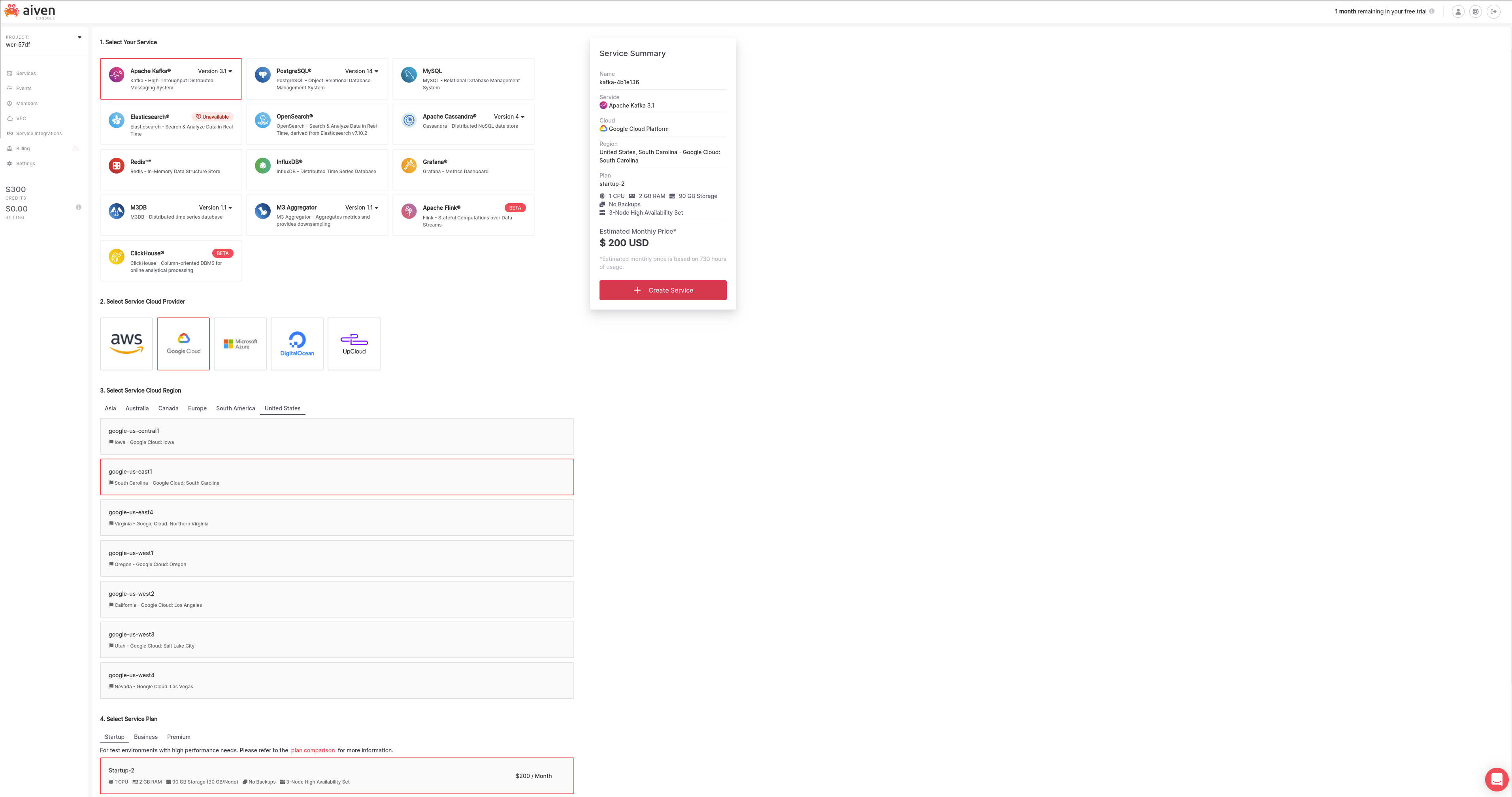
You'll also need to set up a total of three services on the Aiven platform. The Kafka service itself, InfluxDB where Kafka will send its' metric data, and Grafana, which will provide observability. Aiven provides a free one month trial with $300 in credit for all new accounts which is more than enough for our needs. This [link](https://console.aiven.io/signup/email) takes you to the sign-up page. The instructions are very easy to follow. I recommend setting up a username and password rather than a third party service like Google to create your account for simplicity. The platform provides a very simple guide that walks you through some of the features of the account and provides some limited tailoring of the information it presents if you answer some of their questions. The services are as easy to tear down as they are to set up so remember that if you need to you can always delete a service and recreate it.

Once you've created your account we can set up the services.

## Kafka

First, we'll set up a Kafka service. Find and click on the *Services* link at the top of the left navigation bar as shown in the image.



From here we want to click on the *Create a new service* button. From here, click on the Kafka image to select the service, select a cloud provider to host the service (I chose GCP but it really doesn't matter for our purposes), select the appropriate region for your cloud provider, and finally select the service plan (I suggest the smallest "Startup" plan). It should look similar to the image below.

Now just click *Create Service* and wait for the magic to happen. It will take a few minutes for Aiven to complete the provisioning and configuration.

## InfluxDB

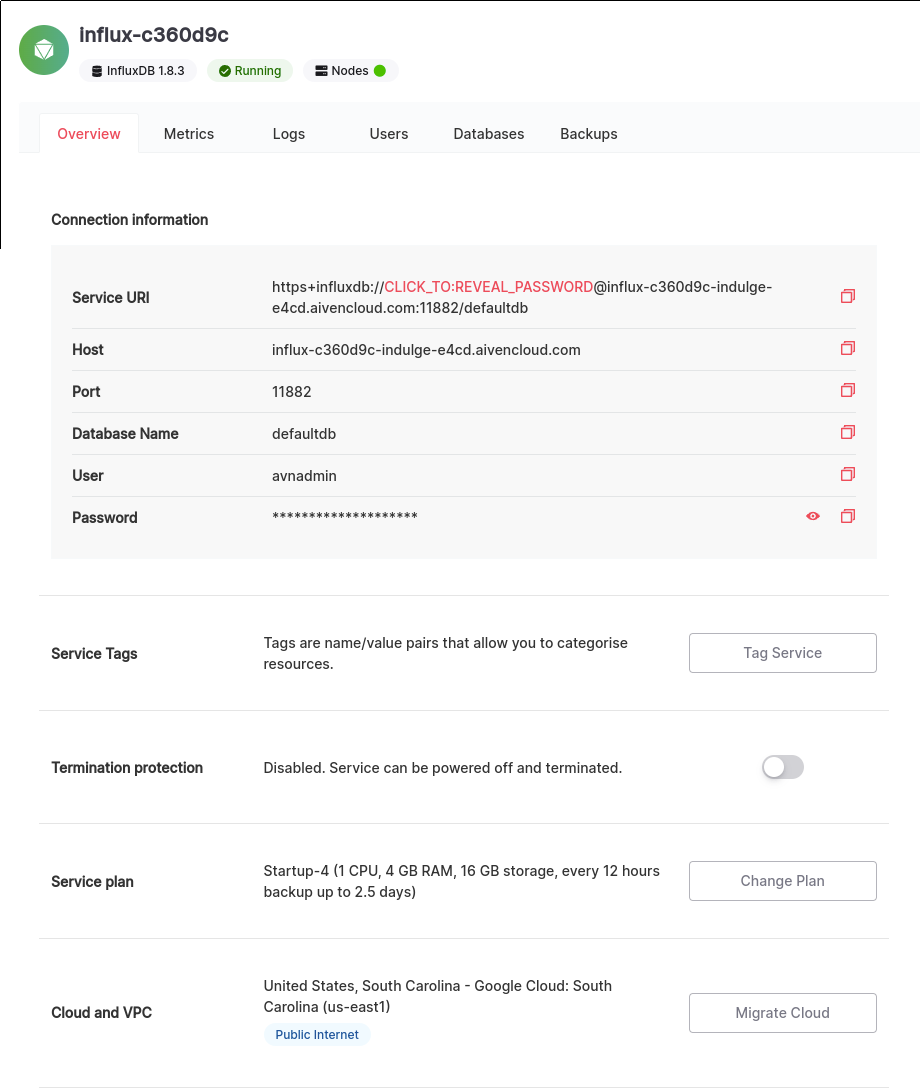
We will be using InfluxDB as the storage place for our Kafka metrics. The process for setting up InfluxDB is similar to that of setting up a Kafka service but instead of selecting the Kafka service icon you will select the InfluxDB service icon. Select the same cloud provider, cloud provider region, and Startup service plan, then click create.

## Grafana

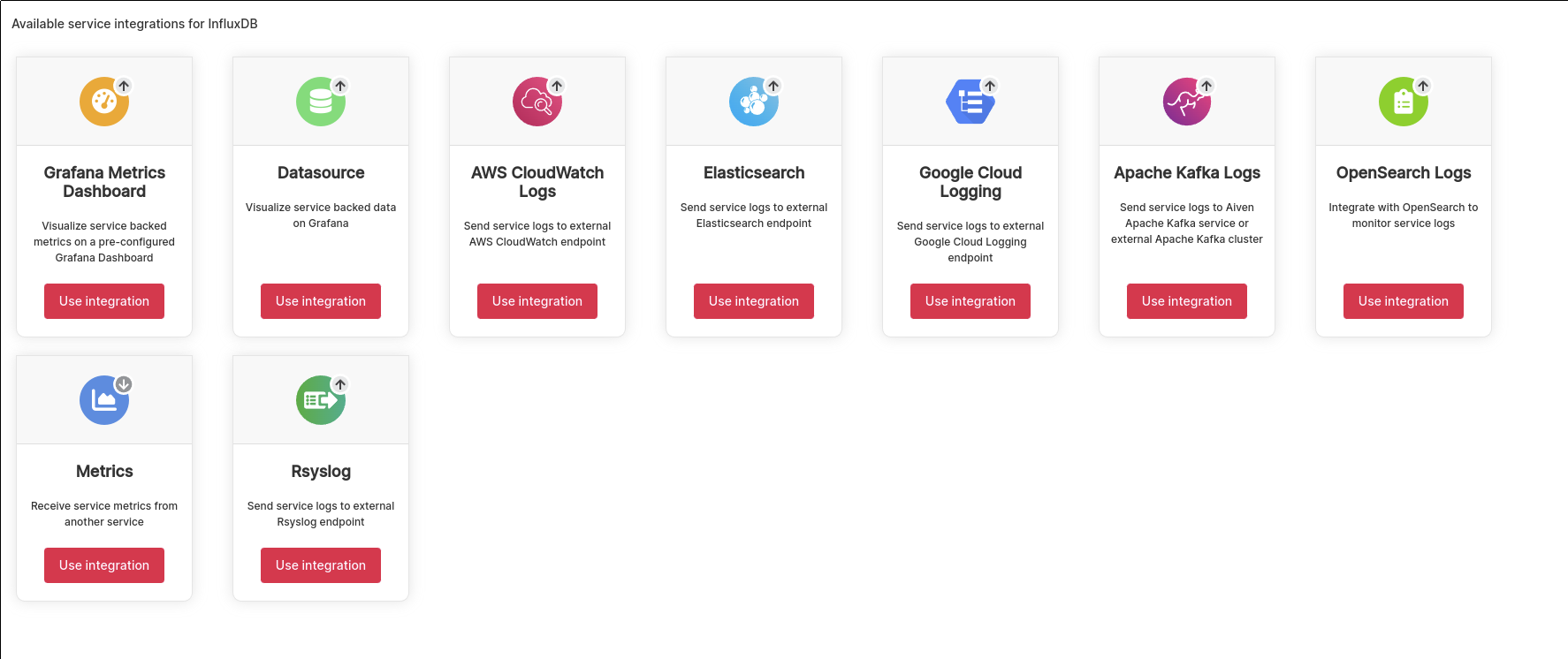
We will be using Grafana as our monitoring tool for the Kafka service. It will read the monitoring data from the InfluxDB service. Repeat the steps to create the previous two services but select the Grafana service icon instead of Kafka or InfluxDB.

## Service Integration

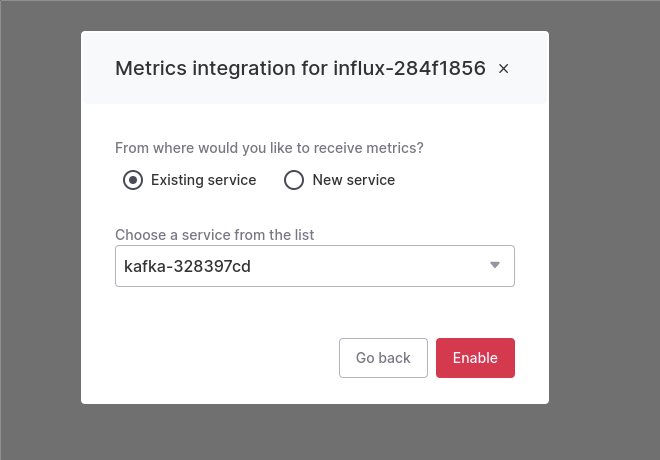
Once all three services have completed deployment (there will be nothing but green circles in the "nodes" column of the services dashboard) click on the row containing InfluxDB. This will take you to the Overview page for the selected service. This page contains the configuration information for the service and will look similar to the image below.



Locate the *Service Integrations* section and click on the *Manage Integrations* button. This brings available integrations for the InfluxDB service. Find the *Metrics* icon on the lower left of the popup and click the *Use integration* button.



The drop-down box that appears will be pre-populated with available services. Select the Kafka service from the drop-down and click *Enable*. The two images below show the integration selection screen and the populated drop-down box.



When this is completed, repeat the steps again to create the Grafana integration. And that's it for integration work!

## Validating Observability

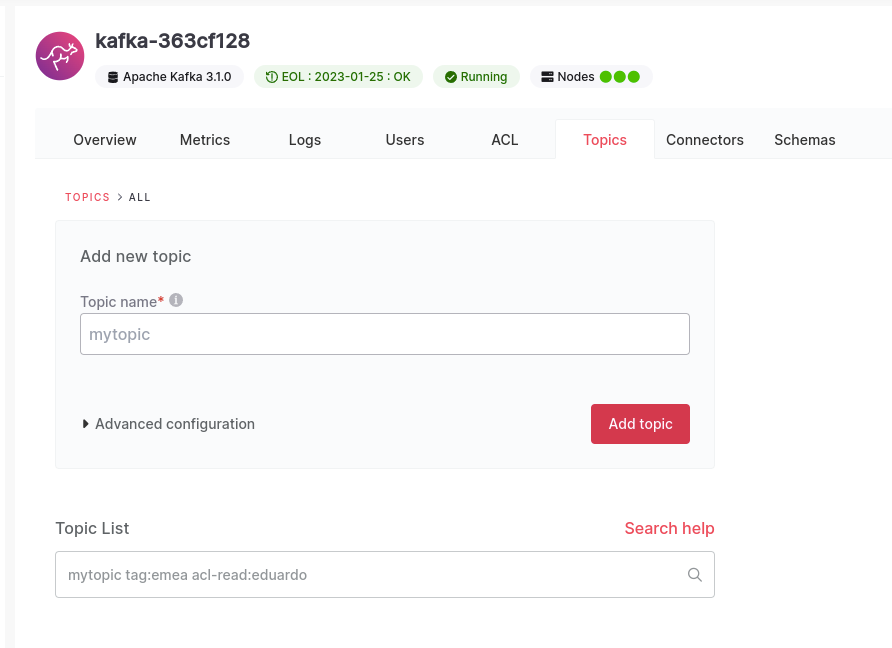
Go to the service overview tab for Grafana and note the service URI, username, and password. Open a browser tab with the address in the Service URI and log into your Grafana service with the username and password. Once you're logged into Grafana, select *Browse* from the dashboard navigation menu on the left navigation pane.



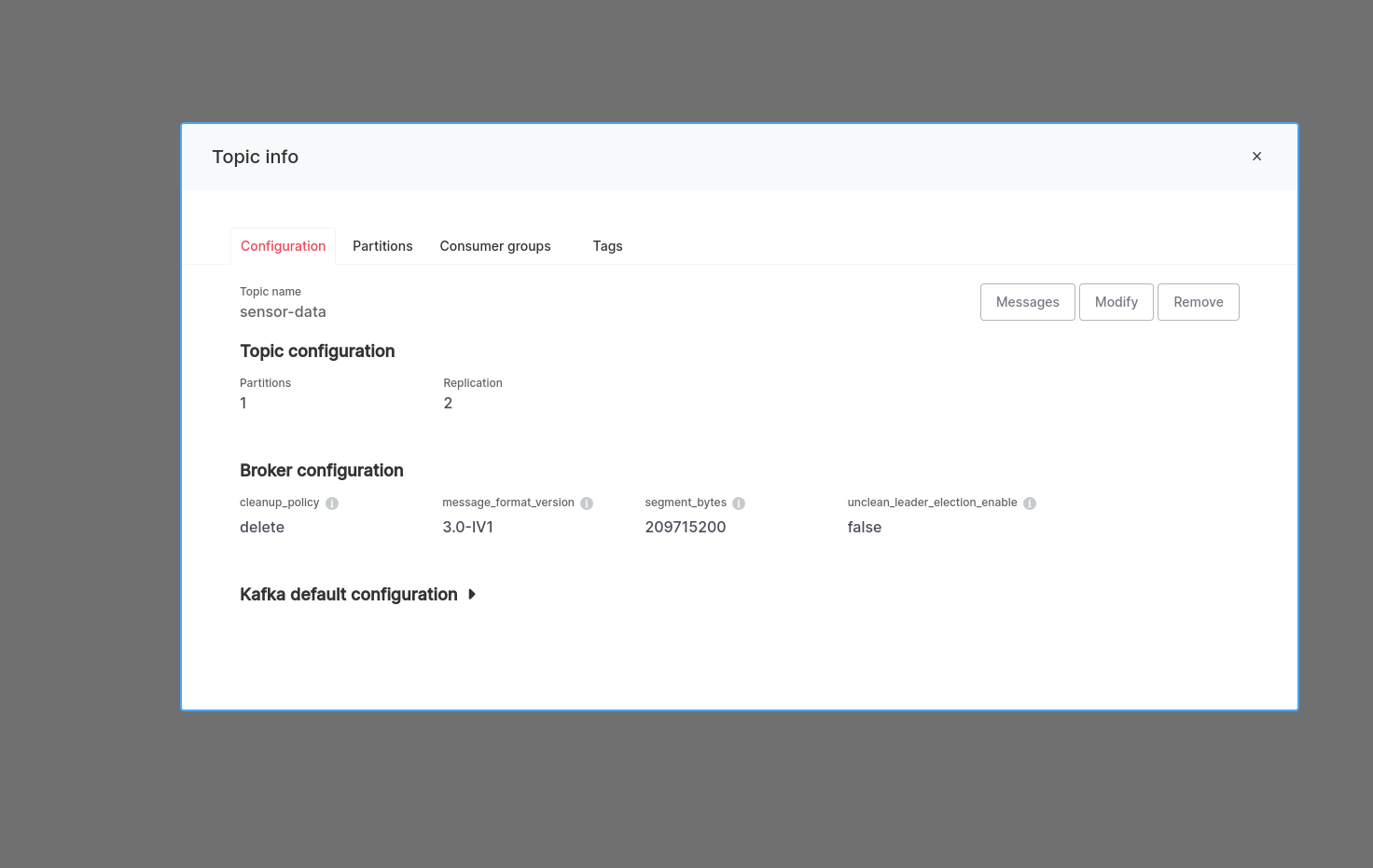
There you will find a pre-configured dashboard. Clicking on the resource will take you to it. Now that we’re at the dashboard we can see lots of pretty charts but how are we certain that it's actually working? Let's check that by submitting events to Kafka and observe the impact on the dashboard.

## Submitting events to Kafka

The simple demo program will emulate an IoT device taking some simple weather data, say an app on a users phone which crowd-sources localized weather data for creating precise local weather maps. In order to submit the events, we will need to be able to communicate with Kafka. Go the the service overview for your Kafka service. You'll want to copy the *Service URI*, and download the *Access Key*, *Access Certificate*, and the *CA Certificate*. These will need to be passed as parameters to the Python script. In addition, we will need a topic to receive the events that we publish. To create the topic, click on the *Topics* tab to the right of the *Overview* tab. Add a new topic called "sensor-data" into the text box and click *Add topic*.



Once the topic has been created you can click on the topic name to create a pop-up that presents detailed topic info for more details.

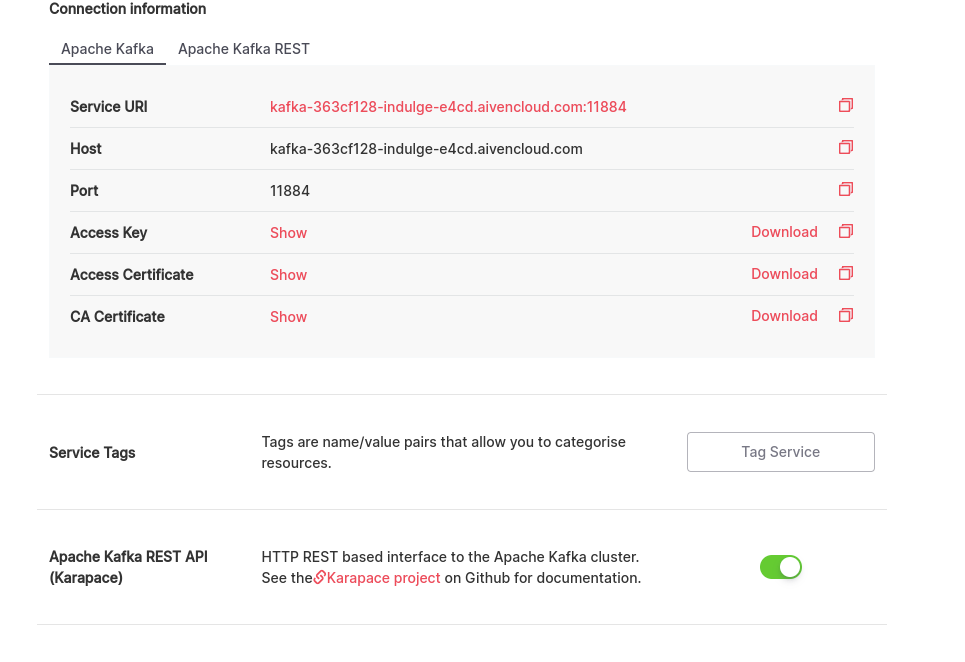


Now, let’s invoke the Python script using the example command below. Be sure to replace the parameters with the data from the service overview page. The message-count parameter determines how many events will be sent to Kafka. Given the speed of the services and the size of our payload, you'll want to set the message-count to at least 100k in order to easily see the impact on the dashboard.

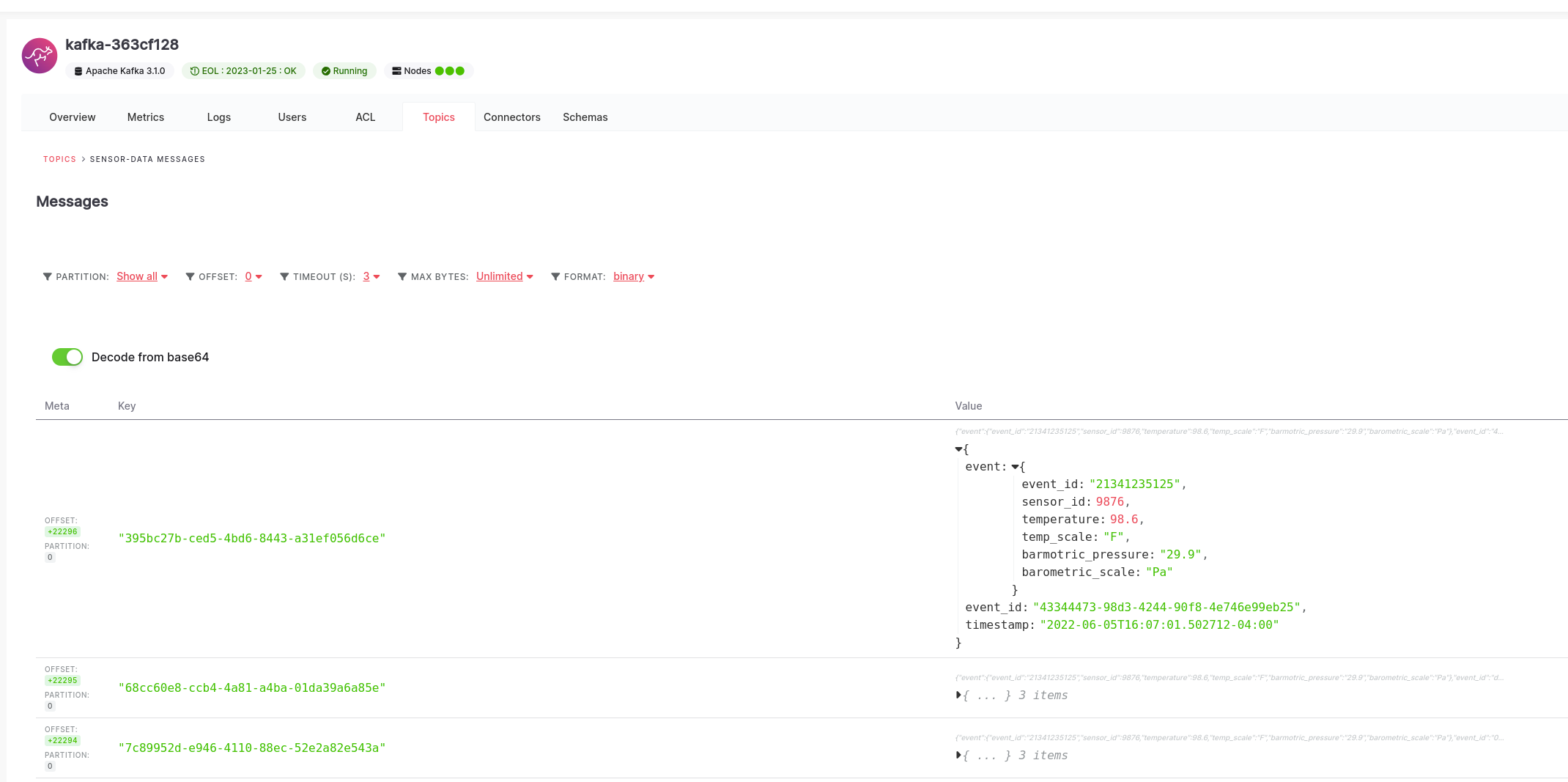
./main.py --service-uri Kafka-service-uri.aivencloud.com:11884 --ca-path path/to/ca.pem --key-path path/to/service.key --cert-path path/to/service.cert --message-count 200000

Once you start sending events you can watch the dashboard as the resource usage increases under the load. If you want to easily look at the messages that the topic has received we can do that as well.

Go to the Service overview tab of the Kafka service and locate the *Apache Kafka REST API* toggle switch and enable it.



Then go back to the topic info pop-up as described above and click on the *Messages* button. After landing on the messages page, click the *Fetch Messages* button. The response may not look like what you expect. Where did the JSON go? By default, the filter is set to return messages in a binary format. To see the actual JSON without having to fetch the messages again, you can simply click the toggle labeled *Decode from base64*, and you can now examine the messages in JSON format.



## Conclusion

This is all that's needed to set up a Kafka service, export metrics, and be able to observe the metrics from a dashboard!