Prototyping Assignment

David Schmidt
Technische Universität Berlin
d.schmidt@campus.tu-berlin.de

Jonas Heisterberg
Technische Universität Berlin
heisterberg@campus.tu-berlin.de

In the following, we will describe our implemented prototype (Fig. 1). Therefore, we will first highlight the deployment architecture, including the deployment environment and the components used. Subsequently, we will propose our communication pattern, which is the core of our implementation.

I. Deployment Architecture

Our Fog Computing application comprises several components and elements to meet the requirements of the levy. The cloud instance is a Google Cloud N2 Standard 4 instance deployed with Terraform. The edge(local) environment consists of a virtual machine running on a local PC. It is provided by Multipass and is intended to simulate a Raspberry Pi with 4GB RAM and 10GB storage. For networking, the cloud node is connected to the node of the local environment via a virtual network using WireGuard (Fig. 2). Data is transferred between the edge and cloud node using the communication pattern described in Section II. The application is written in the hardware-related Golang language and is deployed using a Docker container with Docker Compose. This would be an actual use case, and applications would probably also be deployed based on containerization to ensure portability and security between applications, among other things.

A. Components

Our Fog application comprises four Docker containers, three running in the cloud and one at the edge. The PostgreSQL database container stores client and sensor message data, while the CloudBeaver one allows you to inspect database tables. The Cloud Application container handles the logic to accept messages from the edge, store them in the database, and send analyzed data back to the Edge Application. Finally, the Edge Application container collects sensor data, sends it to the Cloud Application, and receives analyzed data in return (Fig. 2).

The sensor client manages the sensors acting independently of each other at configurable intervals. The virtual sensor simulates temperature and humidity data, while the memory sensor collects data on memory usage. In addition, the CPU sensor collects data on CPU usage. The gathered data is then sent to the cloud and stored in the database, where the memory and virtual data of the last minute are then analyzed for the metrics: average, standard deviation, mean, min, and max.

II. Communication Pattern

Deriving from the requirements, the connection between the two nodes must fulfil the following aspect: Data must be transmitted concurrently between both nodes. Data must be buffered for subsequent transmission when one node disconnects or crashes. Therefore, we implemented a Golang package to abstract the underlying logic. We will provide insights on accomplishing the desired behaviour implemented in the package. Consequently, we will show how our implementation handles communication internally and establishes a connection.

A. Bidirectional Communication and Disconnection Handling

We leverage the full-duplex nature of the TCP connection to transmit data concurrently between the cloud and edge nodes. We employ a specific message structure where content is enhanced with a timestamp and topic. The buffering is integrated by enforcing each node with two First In First Out (FIFO) queues, one responsible for all ingress messages and the other for all egress messages. Sending a message is represented by pushing a message into the egress queue, and receiving is represented by popping an item from the ingress queue. An internal running routine serves each queue go routine. If the connection is open, the receiving routine receives messages from the TCP connection and enqueues them in the ingress queue. The sending routine dequeues all items successively and transmits them using the TCP connection. Both receiving routines will be notified and stopped instantly if the connection is closed, preventing them from further transmission of messages. Consequently, the messages are stored within the queue for subsequent submission. If a disconnection occurs, the connecting node will try to establish a connection (Section II-B) and, if connected, start the go routines again.

B. Establishing a Connection

We divide potential communication nodes into listening and connecting nodes. The difference between both nodes is that the listening nodes await a connect call from the connecting nodes; one listening node can be connected to multiple connecting nodes, using their IP as differentiation. When trying to connect, the connecting node uses an exponential backoff with a maximum timeout delay to prevent failure when the counterpart is offline. If the listener registers a connection with a previously connected node, the queues associated with that node are used to avoid data loss (Fig. 3).

III. Conclusion

The software described in the project successfully demonstrates a robust fog computing application that enables real-time data processing and resilient communication. The application architecture ensures continuous operation and data integrity, even during network interruptions. The deployment of Docker containers in a virtual network, with Golang and TCP communication, shows a possible practical implementation of fog computing principles. In the subsequent slides, we provide screenshots of the running applications.

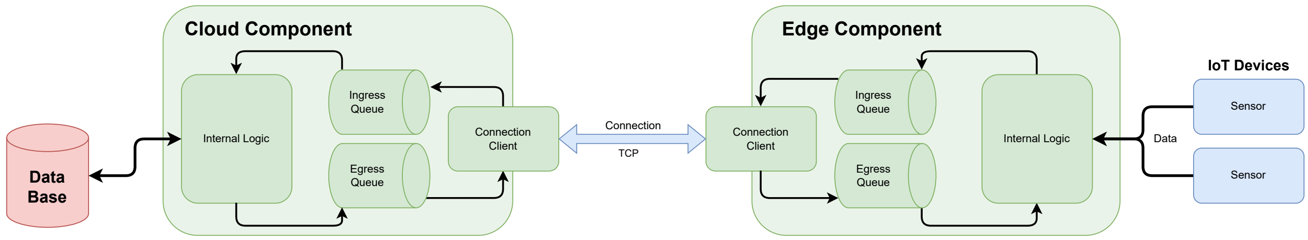


Fig. 1. The system design of the proposed application. The application employs queues for data buffering and leverages a TCP Channel for connection and transmission. The edge component collects data from simulated IoT sensors, while the cloud component processes and stores this data in the database.

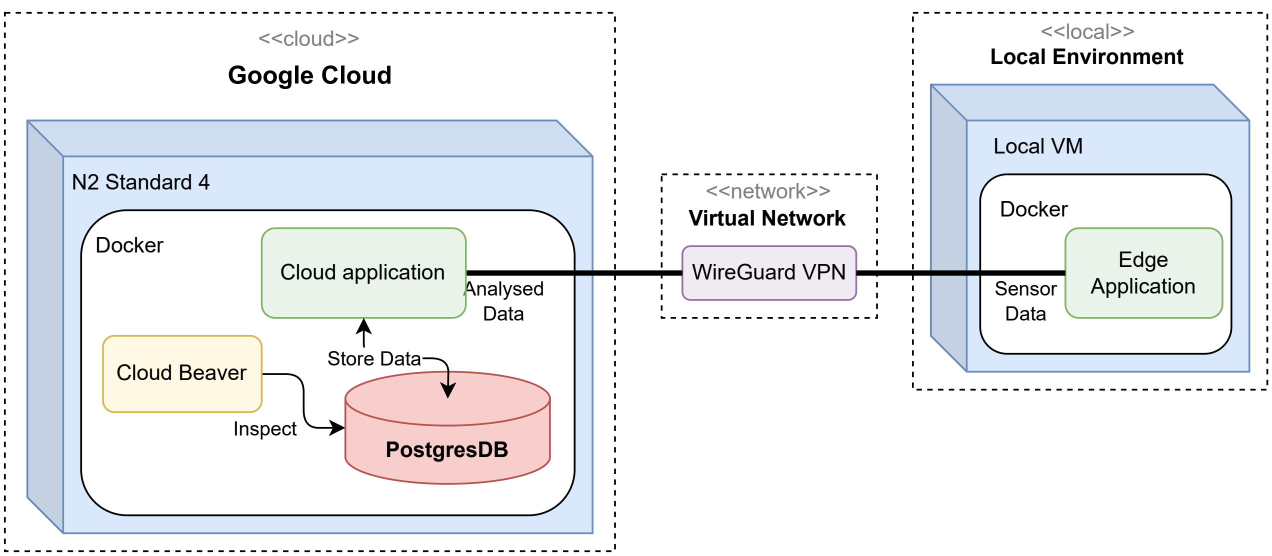


Fig. 2. The deployment architecture of the fog computing application. Cloud application is connected to edge application using WireGuard VPN. Cloud also deploys a Postgres DB for data storage and a Cloud Beaver for inspection.

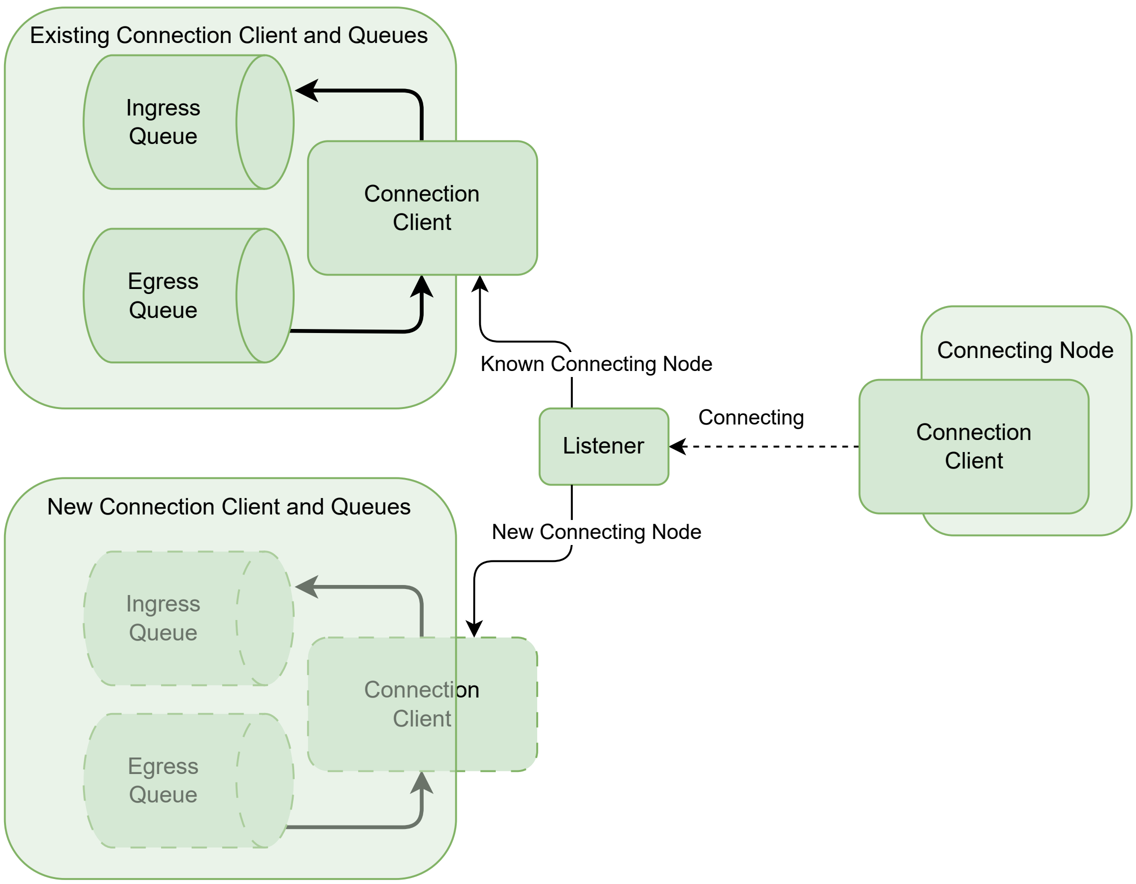


Fig. 3. When a new connecting node tries to connect with the listener, the listener creates a new connection(top). When a node reconnects, the listener attaches the queues associated with that node(Bottom).

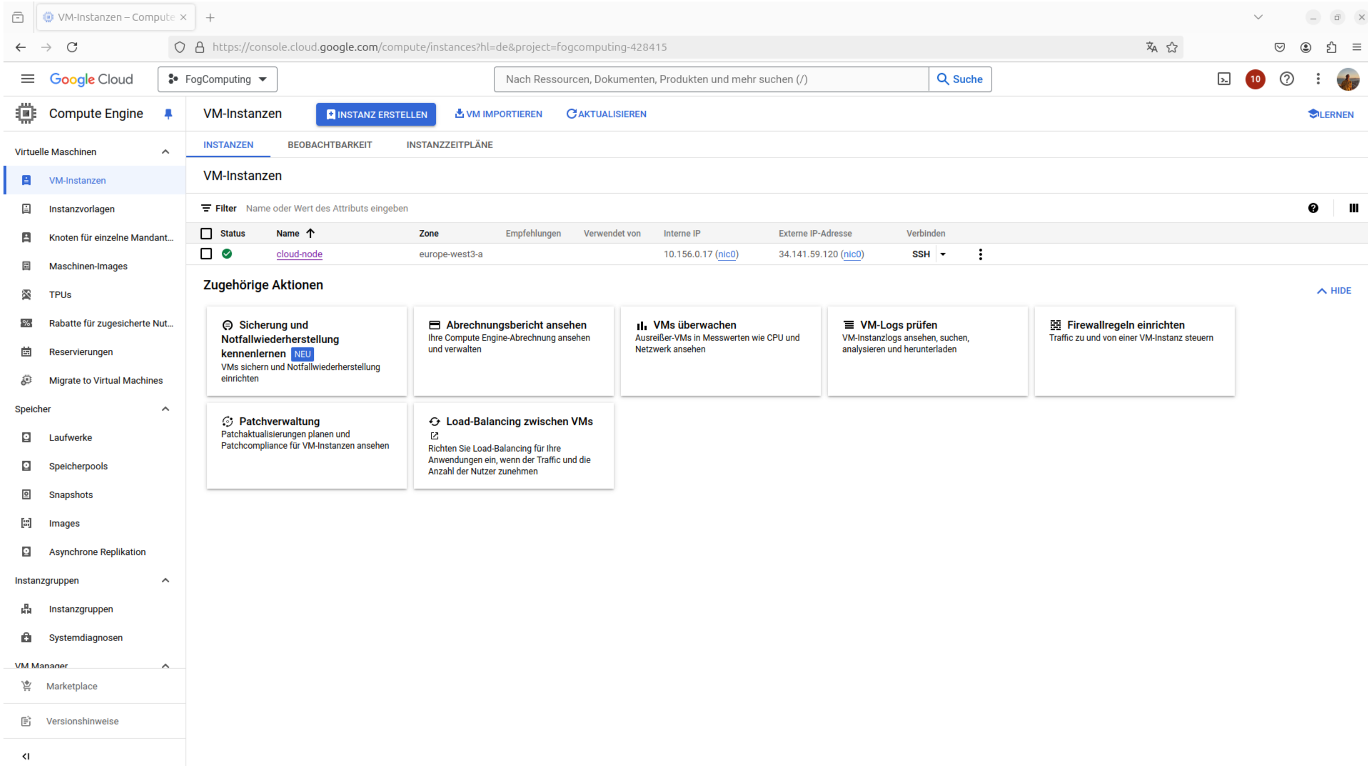


Fig. 4. The cloud node is provisioned by the Google Cloud Platform

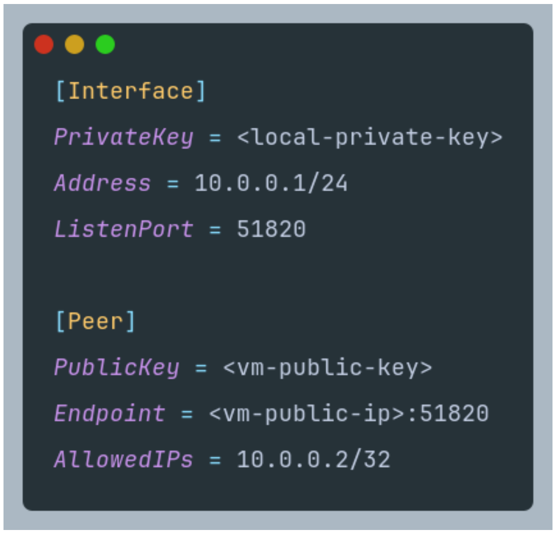


Fig. 5. Wireguard: The configuration file on the local pc

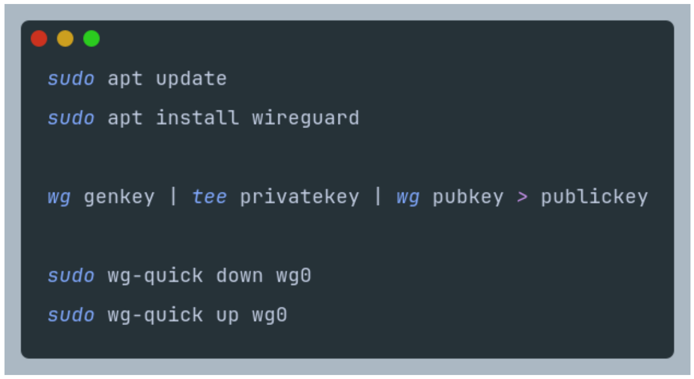


Fig. 6. Wireguard: The cli commands to set up the virtual private network

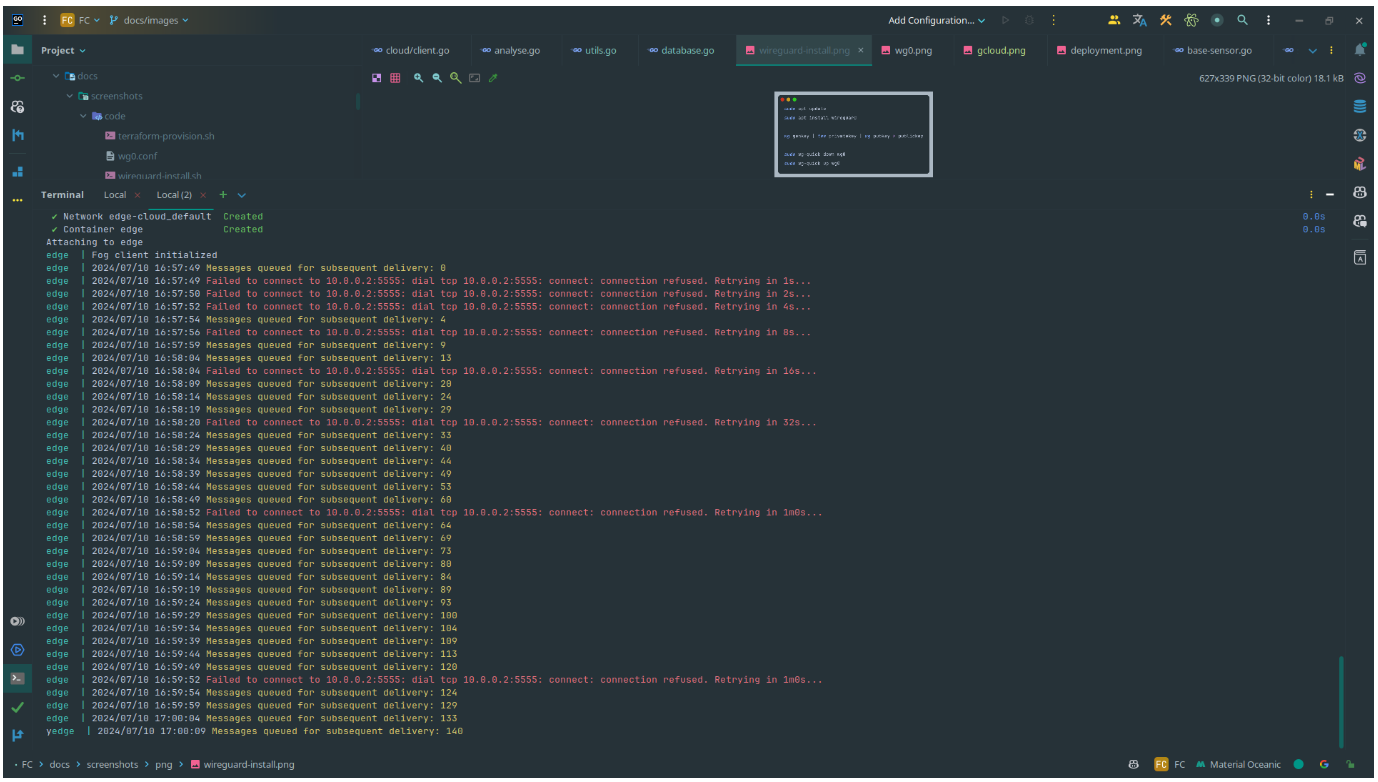


Fig. 7. The edge application is offline and tries to connect to the cloud application

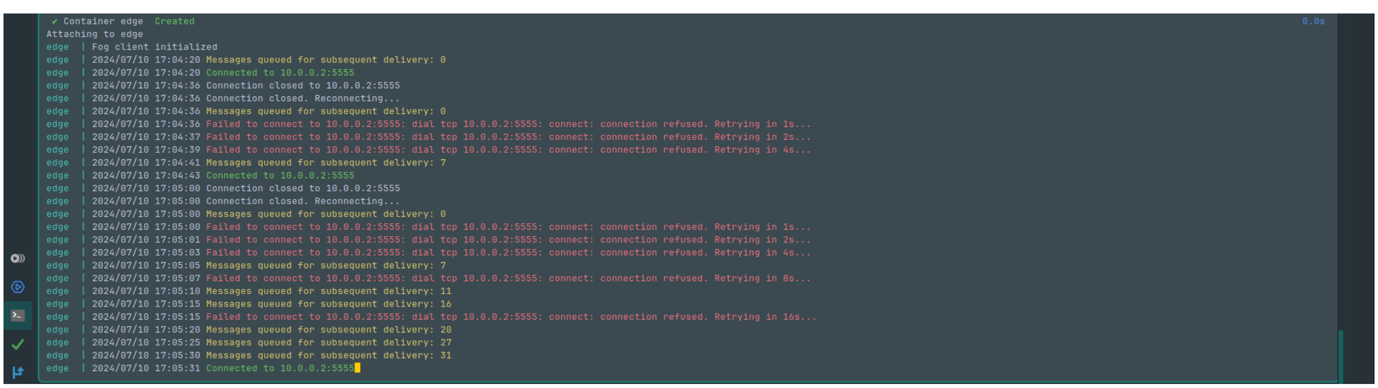


Fig. 8. The edge application has been online but lost connection and tries to reconnect

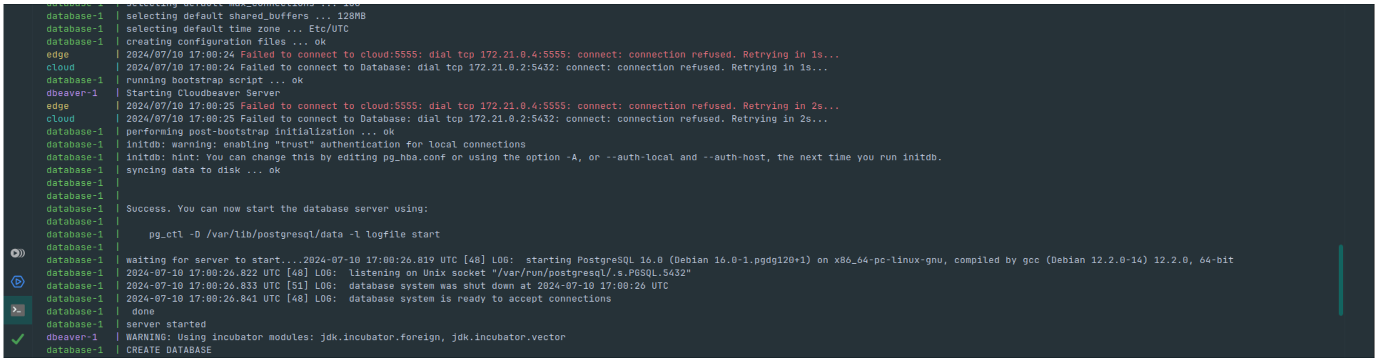


Fig. 9. The cloud starts and tries to connect to the database

Fig. 10. The cloud runs and receives queued sensor messages from the edge application

Fig. 11. The cloud is capable of accepting multiple edge connections.

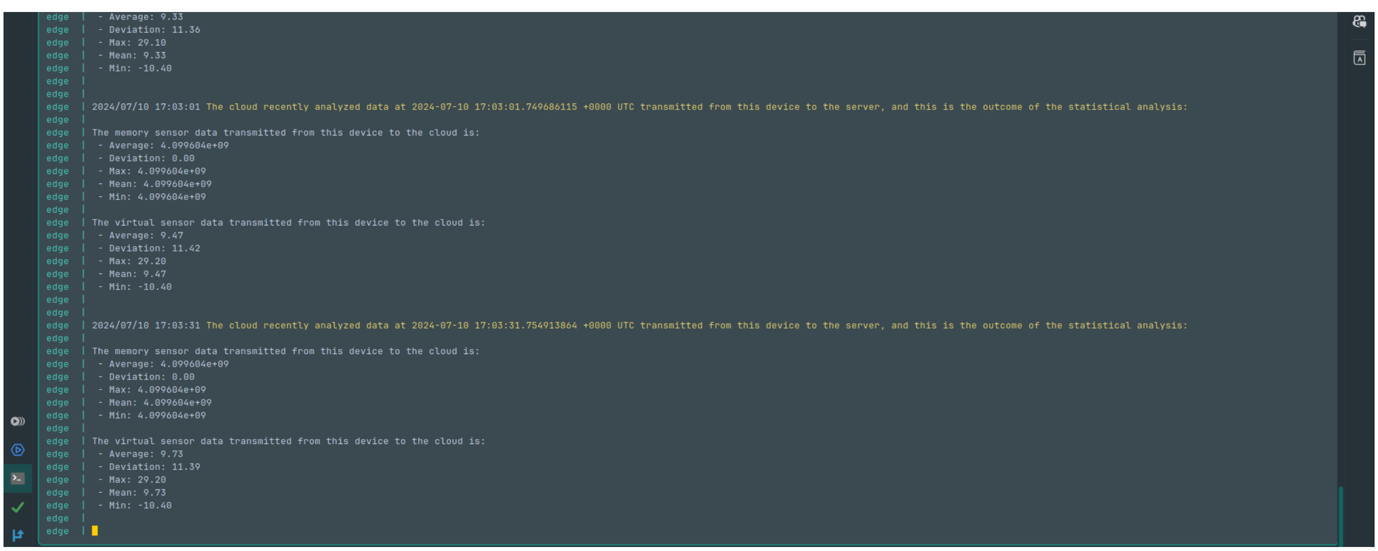


Fig. 12. The edge application on the local machine receives the analysis from the cloud application

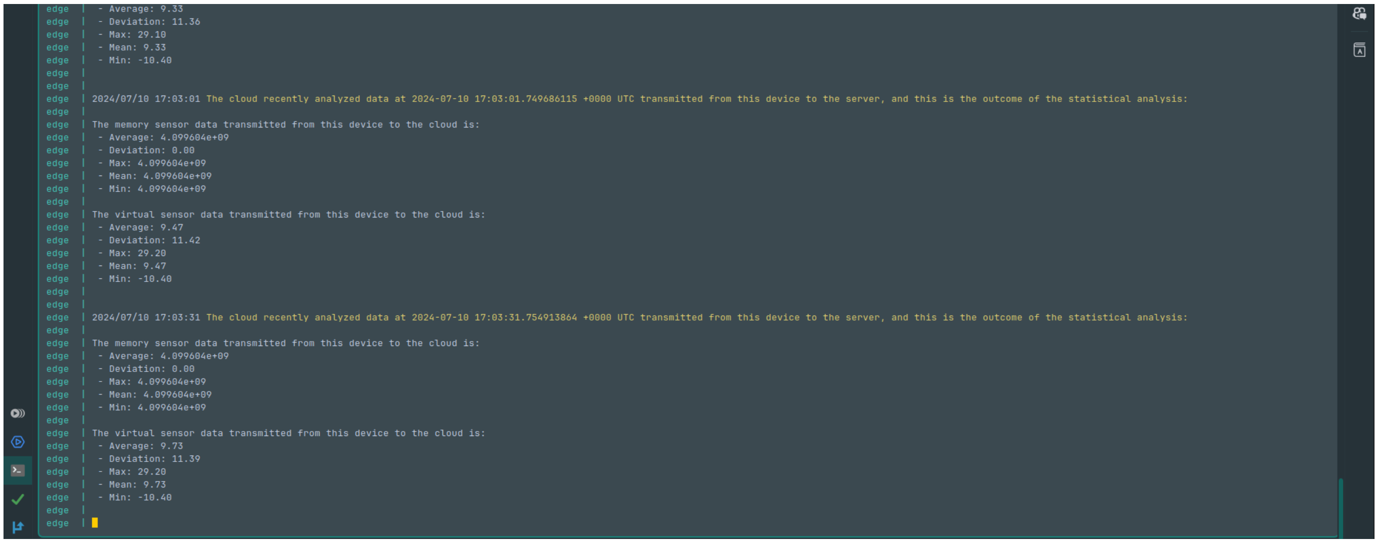


Fig. 13. A possible second edge application receives the analysis from the cloud application

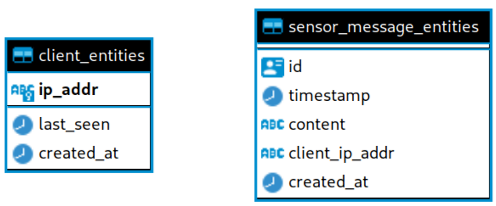


Fig. 14. CloudBeaver: Entity-Relationship-Modell

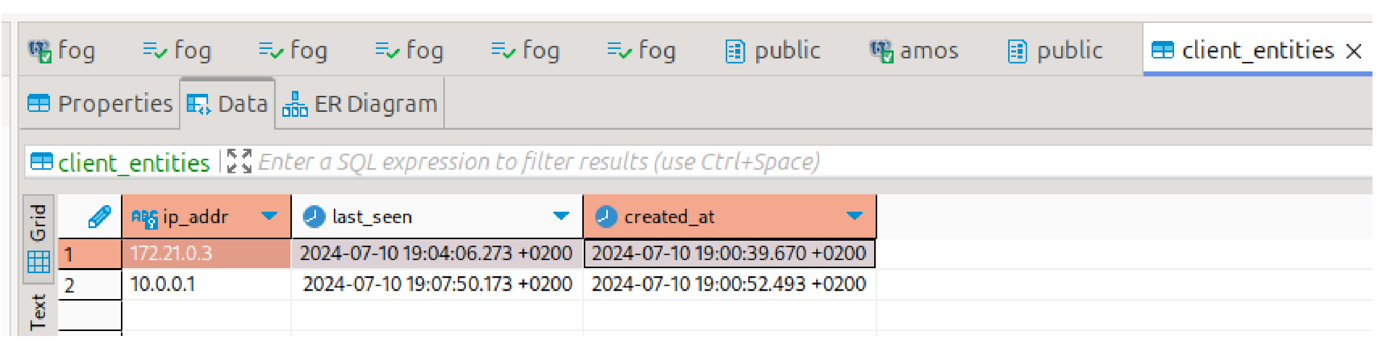
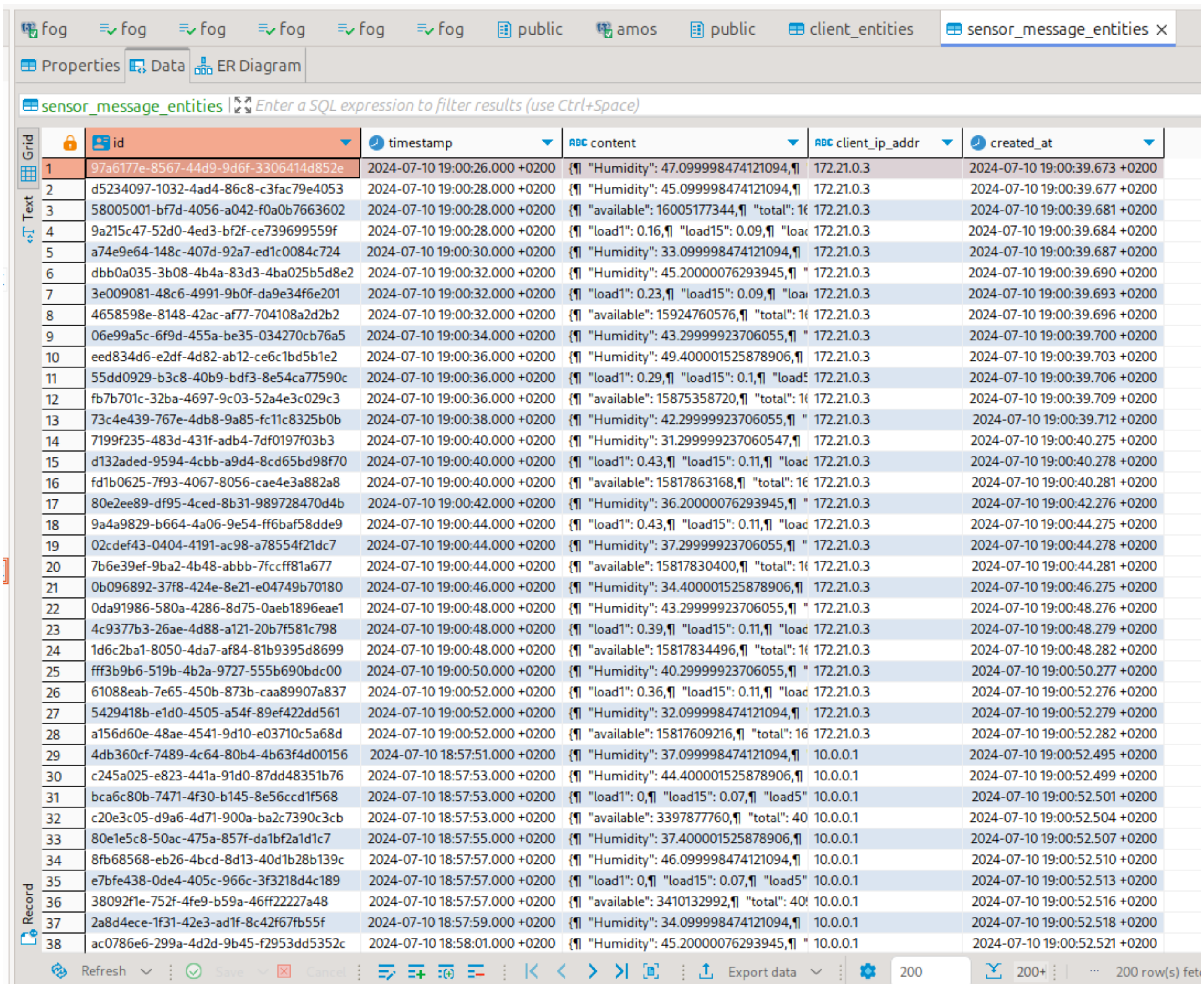


Fig. 15. CloudBeaver: Client entities



	id	timestamp	content	client_ip_addr	created_at
1	97a6177e-8567-44d9-9d6f-3306414d852e	2024-07-10 19:00:26.000 +0200	{¶ "Humidity": 47.099998474121094,¶	172.21.0.3	2024-07-10 19:00:39.673 +0200
2	d5234097-1032-4ad4-86c8-c3fac79e4053	2024-07-10 19:00:28.000 +0200	{¶ "Humidity": 45.099998474121094,¶	172.21.0.3	2024-07-10 19:00:39.677 +0200
3	58005001-bf7d-4056-a042-f0a0b7663602	2024-07-10 19:00:28.000 +0200	{¶ "available": 16005177344,¶ "total": 16	172.21.0.3	2024-07-10 19:00:39.681 +0200
4	9a215c47-52d0-4ed3-bf2f-ce739699559f	2024-07-10 19:00:28.000 +0200	{¶ "load1": 0.16,¶ "load15": 0.09,¶ "loa	172.21.0.3	2024-07-10 19:00:39.684 +0200
5	a74e9e64-148c-407d-92a7-ed1c0084c724	2024-07-10 19:00:30.000 +0200	{¶ "Humidity": 33.099998474121094,¶	172.21.0.3	2024-07-10 19:00:39.687 +0200
6	dbb0a035-3b08-4b4a-83d3-4ba025b5d8e2	2024-07-10 19:00:32.000 +0200	{¶ "Humidity": 45.20000076293945,¶ "	172.21.0.3	2024-07-10 19:00:39.690 +0200
7	3e009081-48c6-4991-9b0f-da9e34f6e201	2024-07-10 19:00:32.000 +0200	{¶ "load1": 0.23,¶ "load15": 0.09,¶ "loa	172.21.0.3	2024-07-10 19:00:39.693 +0200
8	4658598e-8148-42ac-af77-704108a2d2b2	2024-07-10 19:00:32.000 +0200	{¶ "available": 15924760576,¶ "total": 16	172.21.0.3	2024-07-10 19:00:39.696 +0200
9	06e99a5c-6f9d-455a-be35-034270cb76a5	2024-07-10 19:00:34.000 +0200	{¶ "Humidity": 43.29999923706055,¶ "	172.21.0.3	2024-07-10 19:00:39.700 +0200
10	eed834d6-e2df-4d82-ab12-ce6c1bd5b1e2	2024-07-10 19:00:36.000 +0200	{¶ "Humidity": 49.400001525878906,¶	172.21.0.3	2024-07-10 19:00:39.703 +0200
11	55dd0929-b3c8-40b9-bdf3-8e54ca77590c	2024-07-10 19:00:36.000 +0200	{¶ "load1": 0.29,¶ "load15": 0.1,¶ "load5	172.21.0.3	2024-07-10 19:00:39.706 +0200
12	fb7b701c-32ba-4697-9c03-52a4e3c029c3	2024-07-10 19:00:36.000 +0200	{¶ "available": 15875358720,¶ "total": 16	172.21.0.3	2024-07-10 19:00:39.709 +0200
13	73c4e439-767e-4db8-9a85-fc11c8325b0b	2024-07-10 19:00:38.000 +0200	{¶ "Humidity": 42.29999923706055,¶ "	172.21.0.3	2024-07-10 19:00:39.712 +0200
14	7199f235-483d-431f-adb4-7df0197f03b3	2024-07-10 19:00:40.000 +0200	{¶ "Humidity": 31.299999237060547,¶	172.21.0.3	2024-07-10 19:00:40.275 +0200
15	d132aded-9594-4cbb-a9d4-8cd65bd98f70	2024-07-10 19:00:40.000 +0200	{¶ "load1": 0.43,¶ "load15": 0.11,¶ "load	172.21.0.3	2024-07-10 19:00:40.278 +0200
16	fd1b0625-7f93-4067-8056-cae4e3a882a8	2024-07-10 19:00:40.000 +0200	{¶ "available": 15817863168,¶ "total": 16	172.21.0.3	2024-07-10 19:00:40.281 +0200
17	80e2ee89-df95-4ced-8b31-989728470d4b	2024-07-10 19:00:42.000 +0200	{¶ "Humidity": 36.20000076293945,¶ "	172.21.0.3	2024-07-10 19:00:42.276 +0200
18	9a4a9829-b664-4a06-9e54-ff6baf58dde9	2024-07-10 19:00:44.000 +0200	{¶ "load1": 0.43,¶ "load15": 0.11,¶ "load	172.21.0.3	2024-07-10 19:00:44.275 +0200
19	02cdef43-0404-4191-ac98-a78554f21dc7	2024-07-10 19:00:44.000 +0200	{¶ "Humidity": 37.29999923706055,¶ "	172.21.0.3	2024-07-10 19:00:44.278 +0200
20	7b6e39ef-9ba2-4b48-abbb-7fccff81a677	2024-07-10 19:00:44.000 +0200	{¶ "available": 15817830400,¶ "total": 16	172.21.0.3	2024-07-10 19:00:44.281 +0200
21	0b096892-37f8-424e-8e21-e04749b70180	2024-07-10 19:00:46.000 +0200	{¶ "Humidity": 34.400001525878906,¶	172.21.0.3	2024-07-10 19:00:46.275 +0200
22	0da91986-580a-4286-8d75-0aeb1896eae1	2024-07-10 19:00:48.000 +0200	{¶ "Humidity": 43.29999923706055,¶ "	172.21.0.3	2024-07-10 19:00:48.276 +0200
23	4c9377b3-26ae-4d88-a121-20b7f581c798	2024-07-10 19:00:48.000 +0200	{¶ "load1": 0.39,¶ "load15": 0.11,¶ "load	172.21.0.3	2024-07-10 19:00:48.279 +0200
24	1d6c2ba1-8050-4da7-af84-81b9395d8699	2024-07-10 19:00:48.000 +0200	{¶ "available": 15817834496,¶ "total": 16	172.21.0.3	2024-07-10 19:00:48.282 +0200
25	fff3b9b6-519b-4b2a-9727-555b690bdc00	2024-07-10 19:00:50.000 +0200	{¶ "Humidity": 40.29999923706055,¶ "	172.21.0.3	2024-07-10 19:00:50.277 +0200
26	61088eab-7e65-450b-873b-caa89907a837	2024-07-10 19:00:52.000 +0200	{¶ "load1": 0.36,¶ "load15": 0.11,¶ "load	172.21.0.3	2024-07-10 19:00:52.276 +0200
27	5429418b-e1d0-4505-a54f-89ef422dd561	2024-07-10 19:00:52.000 +0200	{¶ "Humidity": 32.099998474121094,¶	172.21.0.3	2024-07-10 19:00:52.279 +0200
28	a156d60e-48ae-4541-9d10-e03710c5a68d	2024-07-10 19:00:52.000 +0200	{¶ "available": 15817609216,¶ "total": 16	172.21.0.3	2024-07-10 19:00:52.282 +0200
29	4db360cf-7489-4c64-80b4-4b63f4d00156	2024-07-10 18:57:51.000 +0200	{¶ "Humidity": 37.099998474121094,¶	10.0.0.1	2024-07-10 19:00:52.495 +0200
30	c245a025-e823-441a-91d0-87dd48351b76	2024-07-10 18:57:53.000 +0200	{¶ "Humidity": 44.400001525878906,¶	10.0.0.1	2024-07-10 19:00:52.499 +0200
31	bca6c80b-7471-4f30-b145-8e56ccd1f568	2024-07-10 18:57:53.000 +0200	{¶ "load1": 0,¶ "load15": 0.07,¶ "load5"	10.0.0.1	2024-07-10 19:00:52.501 +0200
32	c20e3c05-d9a6-4d71-900a-ba2c7390c3cb	2024-07-10 18:57:53.000 +0200	{¶ "available": 3397877760,¶ "total": 40	10.0.0.1	2024-07-10 19:00:52.504 +0200
33	80e1e5c8-50ac-475a-857f-da1bf2a1d1c7	2024-07-10 18:57:55.000 +0200	{¶ "Humidity": 37.400001525878906,¶	10.0.0.1	2024-07-10 19:00:52.507 +0200
34	8fb68568-eb26-4bcd-8d13-40d1b28b139c	2024-07-10 18:57:57.000 +0200	{¶ "Humidity": 46.099998474121094,¶	10.0.0.1	2024-07-10 19:00:52.510 +0200
35	e7bfe438-0de4-405c-966c-3f3218d4c189	2024-07-10 18:57:57.000 +0200	{¶ "load1": 0,¶ "load15": 0.07,¶ "load5"	10.0.0.1	2024-07-10 19:00:52.513 +0200
36	38092f1e-752f-4fe9-b59a-46ff22227a48	2024-07-10 18:57:57.000 +0200	{¶ "available": 3410132992,¶ "total": 40	10.0.0.1	2024-07-10 19:00:52.516 +0200
37	2a8d4ece-1f31-42e3-ad1f-8c42f67fb55f	2024-07-10 18:57:59.000 +0200	{¶ "Humidity": 34.099998474121094,¶	10.0.0.1	2024-07-10 19:00:52.518 +0200
38	ac0786e6-299a-4d2d-9b45-f2953dd5352c	2024-07-10 18:58:01.000 +0200	{¶ "Humidity": 45.20000076293945,¶ "	10.0.0.1	2024-07-10 19:00:52.521 +0200

Fig. 16. CloudBeaver: Sensor message entities