

Test Scenarios

1 - Emit SIGHUP

expected results

The program captures that signal and checks if there are any changes for the Monitor or the Server to reload the configuration.

- open a terminal and run program:

```
roemer@ubuntuserver:~/C++/QuartuxProject$ ./bin/QMonitor
{"level":"INFO","msg":"start Monitor"}
{"level":"INFO","msg":"service started"}
```

- open other terminal and send a SIGHUP signal, getting PID

```
roemer@ubuntuserver:~$ ps -aux | grep QMonitor
roemer    124916  0.0  0.2 234756 12332 pts/4    Sl+  01:21   0:00 ./bin/QMonitor
roemer    124944  0.0  0.0   9228  2516 pts/5    S+   01:23   0:00 grep --color=auto QMonitor
roemer@ubuntuserver:~$
```

PID is 124916

emit signal SIGHUP

```
roemer@ubuntuserver:~$ kill -HUP 124916
roemer@ubuntuserver:~$
```

first terminal

```
roemer@ubuntuserver:~/C++/QuartuxProject$ ./bin/QMonitor
{"level":"INFO","msg":"start Monitor"}
{"level":"INFO","msg":"service started"}
{"level":"INFO","msg":"signal SIGHUP received"}
█
```

logs

```
roemer@ubuntu:~$ tail -f /var/log/server_service.log
{"timestamp":"2025-10-01T22:24:05","level":"INFO","msg":"stop Server waiting for thread"}
{"timestamp":"2025-10-01T23:19:06","level":"INFO","msg":"start Server"}
{"timestamp":"2025-10-01T23:19:09","level":"INFO","msg":"stop Server waiting for thread"}
{"timestamp":"2025-10-02T00:46:28","level":"INFO","msg":"start Server"}
{"timestamp":"2025-10-02T00:46:50","level":"INFO","msg":"status request received"}
{"timestamp":"2025-10-02T00:47:47","level":"INFO","msg":"status request received"}
{"timestamp":"2025-10-02T00:49:40","level":"INFO","msg":"status request received"}
{"timestamp":"2025-10-02T00:50:25","level":"INFO","msg":"status request received"}
{"timestamp":"2025-10-02T01:14:16","level":"INFO","msg":"stop Server waiting for thread"}
{"timestamp":"2025-10-02T01:21:40","level":"INFO","msg":"start Server"}
```

in this case reload configuration was not made because any value of the initial configuration was made

open file config/config.json and change listening port for example from 'rest_port=8090' to 'rest_port=8080' save it and emit the same signal

now a new records was displayed in server_service.log indicating that a reload configuration was performed

```
roemer@ubuntu:~$ tail -f /var/log/server_service.log
{"timestamp":"2025-10-02T00:46:50","level":"INFO","msg":"status request received"}
{"timestamp":"2025-10-02T00:47:47","level":"INFO","msg":"status request received"}
{"timestamp":"2025-10-02T00:49:40","level":"INFO","msg":"status request received"}
{"timestamp":"2025-10-02T00:50:25","level":"INFO","msg":"status request received"}
{"timestamp":"2025-10-02T01:14:16","level":"INFO","msg":"stop Server waiting for thread"}
{"timestamp":"2025-10-02T01:21:40","level":"INFO","msg":"start Server"}
{"timestamp":"2025-10-02T01:36:36","level":"INFO","msg":"setting a new port"}
{"timestamp":"2025-10-02T01:36:36","level":"INFO","msg":"reload configuration Server"}
{"timestamp":"2025-10-02T01:36:36","level":"INFO","msg":"stop Server waiting for thread"}
{"timestamp":"2025-10-02T01:36:36","level":"INFO","msg":"start Server"}
```

for Monitor was not made changes and y doesnt perform reload configuration

open file config/config.json and change interval value for example 'query_interval=10' to 'query_interval=20' saved and emit the same signal

now new records are displayed in monitor_service.log indicating that reload configuration was performed

```
{"timestamp":"2025-10-02T01:52:46","level":"INFO","msg":"3 Humidity 30 2025-09-26 13:38:49"}
{"timestamp":"2025-10-02T01:52:46","level":"INFO","msg":"processed 3 rows."}
{"timestamp":"2025-10-02T01:52:58","level":"INFO","msg":"reloading configuration"}
{"timestamp":"2025-10-02T01:52:58","level":"INFO","msg":"stop Monitor waiting for thread"}
{"timestamp":"2025-10-02T01:52:58","level":"INFO","msg":"start Monitor"}
{"timestamp":"2025-10-02T01:52:58","level":"INFO","msg":"1 Temperature 42 2025-09-26 13:38:49"}
{"timestamp":"2025-10-02T01:52:58","level":"INFO","msg":"2 Pressure 60 2025-09-26 13:38:49"}
{"timestamp":"2025-10-02T01:52:58","level":"WARNING","msg":"threshold exceeded"}
{"timestamp":"2025-10-02T01:52:58","level":"INFO","msg":"3 Humidity 30 2025-09-26 13:38:49"}
{"timestamp":"2025-10-02T01:52:58","level":"INFO","msg":"processed 3 rows."}
{"timestamp":"2025-10-02T01:53:08","level":"INFO","msg":"1 Temperature 42 2025-09-26 13:38:49"}
{"timestamp":"2025-10-02T01:53:08","level":"INFO","msg":"2 Pressure 60 2025-09-26 13:38:49"}
{"timestamp":"2025-10-02T01:53:08","level":"WARNING","msg":"threshold exceeded"}
{"timestamp":"2025-10-02T01:53:08","level":"INFO","msg":"3 Humidity 30 2025-09-26 13:38:49"}
{"timestamp":"2025-10-02T01:53:08","level":"INFO","msg":"processed 3 rows."}
```

2 - Emit SIGINT

expected results

program catch that signal and closes the program in an orderly manner

open a terminal and emit SIGINT signal

```
roemer@ubuntuuser:~$ ps -aux | grep QMonitor
roemer  125617  0.0  0.2 234900 12384 pts/4    Sl+  01:51   0:00 ./bin/QMonitor
roemer  125701  0.0  0.0   9228  2520 pts/5    S+   02:01   0:00 grep --color=auto QMonitor
roemer@ubuntuuser:~$ kill -SIGINT 125617
roemer@ubuntuuser:~$
```

first terminal display that SIGINT signal was catch and close everything

```
roemer@ubuntuuser:~/C++/QuartuxProject$ ./bin/QMonitor
{"level":"INFO","msg":"service started"}
{"level":"INFO","msg":"signal SIGHUP received"}
{"level":"INFO","msg":"signal SIGINT received"}
{"level":"INFO","msg":"shutting down"}
roemer@ubuntuuser:~/C++/QuartuxProject$
```

in monitor_service.log is displayed

```
{"timestamp":"2025-10-02T02:01:44","level":"INFO","msg":"processed 3 rows."}
{"timestamp":"2025-10-02T02:01:51","level":"INFO","msg":"stop Monitor waiting for thread"}
```

in server_service.log is displayed

```
{"timestamp":"2025-10-02T01:51:26","level":"INFO","msg":"start Server"}
{"timestamp":"2025-10-02T02:01:51","level":"INFO","msg":"stop Server waiting for thread"}
```

3 - Request status Monitor endpoint

expected result

a json with this forma:

```
{
  "last_check": "2025-07-08T10:23:00Z",
  "rows_processed": 3,
  "alerts": 1
}
```

open a terminal 1, run bin/QMonitor

```
roemer@ubuntuuser:~/C++/QuartuxProject$ ./bin/QMonitor
{"level":"INFO","msg":"service started"}
```

open other terminal 2 and cature package tcp-ip at port 8080

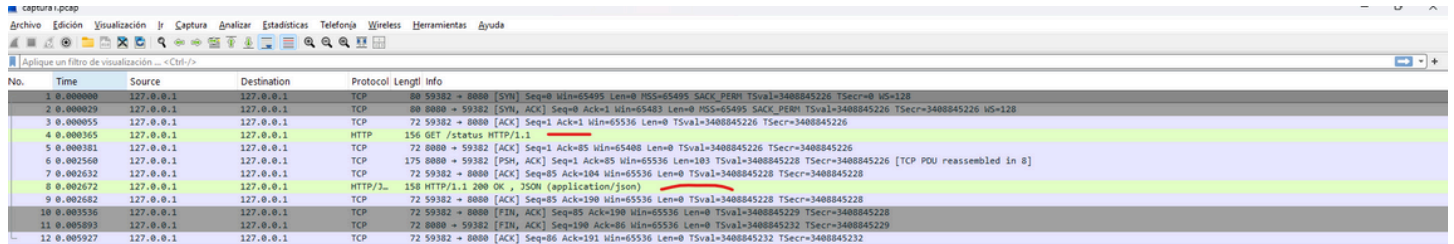
```
roemer@ubuntuuser:~$ sudo tcpdump -s0 -w capture.pcap -i any tcp port 8080
[sudo] password for roemer:
tcpdump: data link type LINUX_SLL2
tcpdump: listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
```

open other terminal 3 execute curl command tu request status endpoint

```
roemer@ubuntuuserver:~$ curl http://127.0.0.1:8080/status
{
  "alerts": 1,
  "last_check": "2025-10-02T03:00:21Z",
  "rows_processed": 3
}
roemer@ubuntuuserver:~$
```

stop tcpdump in terminal 2

move the file generated with tcpdump captura1.pcap to windows SO and open it with wireshark.
The traffic at port 8080 is displayed



The image shows a Wireshark packet capture of an HTTP GET request to /status. The packet list shows a sequence of packets: a SYN packet, an ACK packet, a GET packet, and several ACK packets. The packet details pane shows the GET packet structure, including the status line 'HTTP/1.1 200 OK', the content type 'application/json', and the body of the JSON response.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	80	59382 → 8080 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=3408845226 TSecr=0 WS=128
2	0.000029	127.0.0.1	127.0.0.1	TCP	80	8080 → 59382 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=3408845226 TSecr=3408845226 WS=128
3	0.000055	127.0.0.1	127.0.0.1	TCP	72	59382 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=3408845226 TSecr=3408845226
4	0.000365	127.0.0.1	127.0.0.1	HTTP	156	GET /status HTTP/1.1
5	0.000381	127.0.0.1	127.0.0.1	TCP	72	8080 → 59382 [ACK] Seq=1 Ack=85 Win=65480 Len=0 TSval=3408845226 TSecr=3408845226
6	0.002560	127.0.0.1	127.0.0.1	TCP	175	8080 → 59382 [PSH, ACK] Seq=1 Ack=85 Win=65536 Len=103 TSval=3408845228 TSecr=3408845226 [TCP PDU reassembled in 8]
7	0.002632	127.0.0.1	127.0.0.1	TCP	72	59382 → 8080 [ACK] Seq=85 Ack=104 Win=65536 Len=0 TSval=3408845228 TSecr=3408845228
8	0.002672	127.0.0.1	127.0.0.1	HTTP	158	HTTP/1.1 200 OK, JSON (application/json)
9	0.002682	127.0.0.1	127.0.0.1	TCP	72	59382 → 8080 [ACK] Seq=85 Ack=100 Win=65536 Len=0 TSval=3408845228 TSecr=3408845228
10	0.003536	127.0.0.1	127.0.0.1	TCP	72	59382 → 8080 [FIN, ACK] Seq=85 Ack=190 Win=65536 Len=0 TSval=3408845229 TSecr=3408845228
11	0.005893	127.0.0.1	127.0.0.1	TCP	72	8080 → 59382 [FIN, ACK] Seq=190 Ack=86 Win=65536 Len=0 TSval=3408845232 TSecr=3408845229
12	0.005927	127.0.0.1	127.0.0.1	TCP	72	59382 → 8080 [ACK] Seq=86 Ack=191 Win=65536 Len=0 TSval=3408845232 TSecr=3408845232

in server_service.log a record is displayed related with this request

```
{ "timestamp": "2025-10-02T02:54:49", "level": "INFO", "msg": "start Server" }
{ "timestamp": "2025-10-02T02:55:53", "level": "INFO", "msg": "status request received" }
{ "timestamp": "2025-10-02T03:00:24", "level": "INFO", "msg": "status request received" }
```

4 - Request for metrics table every interval seconds

expected result

some records are added to monitor_service.log for every query made to the metrics table with the result of proecessing this information in json format

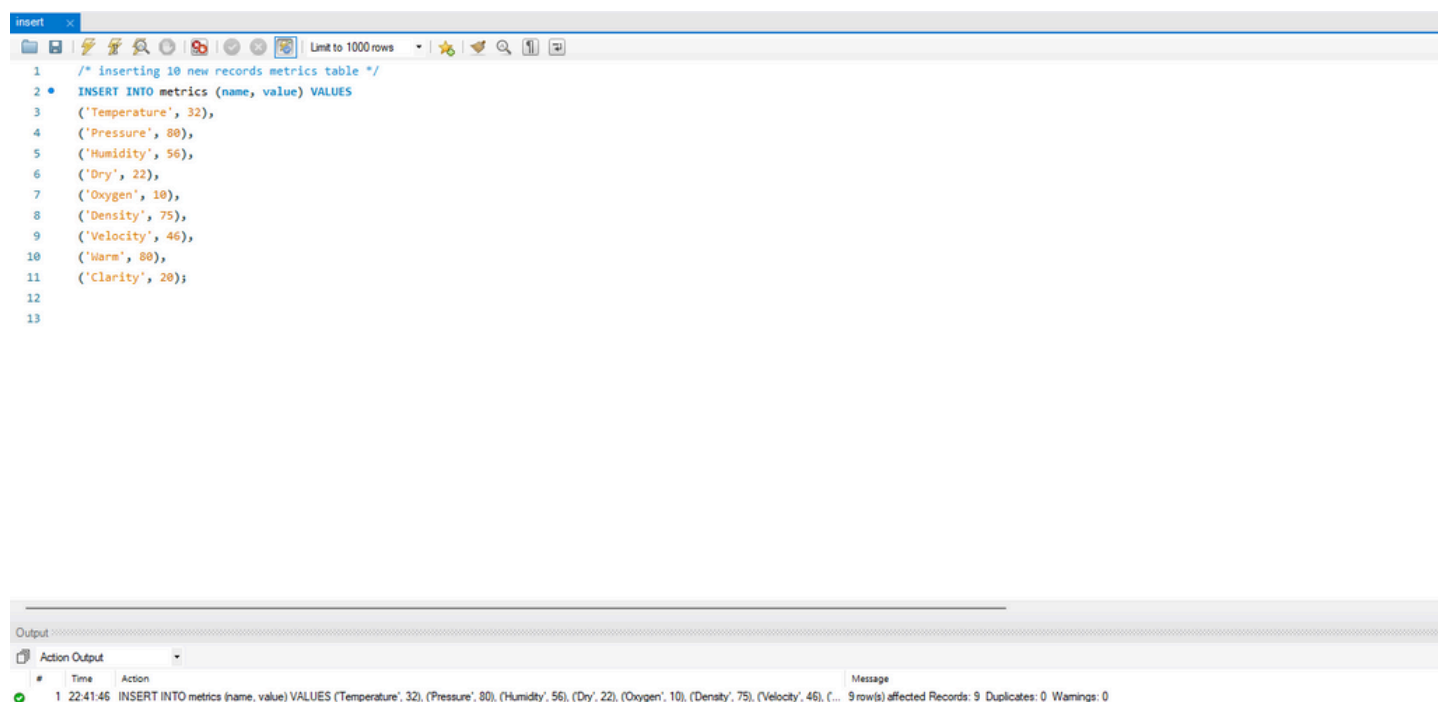
open a terminal 1, run bin/QMonitor

```
roemer@ubuntuuserver:~/C++/QuartuxProject$ ./bin/QMonitor
{"level": "INFO", "msg": "service started"}
```

verify the log file for Monitor every interval seconds new records are added

```
roemer@ubuntuuserver:~$ tail -f /var/log/monitor_service.log
{"timestamp": "2025-10-02T04:38:19", "level": "INFO", "msg": "1 Temperature 42 2025-09-26 13:38:49"}
{"timestamp": "2025-10-02T04:38:19", "level": "INFO", "msg": "2 Pressure 60 2025-09-26 13:38:49"}
{"timestamp": "2025-10-02T04:38:19", "level": "WARNING", "msg": "threshold exceeded"}
{"timestamp": "2025-10-02T04:38:19", "level": "INFO", "msg": "3 Humidity 30 2025-09-26 13:38:49"}
{"timestamp": "2025-10-02T04:38:19", "level": "INFO", "msg": "processed 3 rows."}
{"timestamp": "2025-10-02T04:38:29", "level": "INFO", "msg": "1 Temperature 42 2025-09-26 13:38:49"}
{"timestamp": "2025-10-02T04:38:29", "level": "INFO", "msg": "2 Pressure 60 2025-09-26 13:38:49"}
{"timestamp": "2025-10-02T04:38:29", "level": "WARNING", "msg": "threshold exceeded"}
{"timestamp": "2025-10-02T04:38:29", "level": "INFO", "msg": "3 Humidity 30 2025-09-26 13:38:49"}
{"timestamp": "2025-10-02T04:38:29", "level": "INFO", "msg": "processed 3 rows."}
{"timestamp": "2025-10-02T04:38:39", "level": "INFO", "msg": "1 Temperature 42 2025-09-26 13:38:49"}
{"timestamp": "2025-10-02T04:38:39", "level": "INFO", "msg": "2 Pressure 60 2025-09-26 13:38:49"}
{"timestamp": "2025-10-02T04:38:39", "level": "WARNING", "msg": "threshold exceeded"}
{"timestamp": "2025-10-02T04:38:39", "level": "INFO", "msg": "3 Humidity 30 2025-09-26 13:38:49"}
{"timestamp": "2025-10-02T04:38:39", "level": "INFO", "msg": "processed 3 rows."}
{"timestamp": "2025-10-02T04:38:50", "level": "INFO", "msg": "1 Temperature 42 2025-09-26 13:38:49"}
{"timestamp": "2025-10-02T04:38:50", "level": "INFO", "msg": "2 Pressure 60 2025-09-26 13:38:49"}
{"timestamp": "2025-10-02T04:38:50", "level": "WARNING", "msg": "threshold exceeded"}
{"timestamp": "2025-10-02T04:38:50", "level": "INFO", "msg": "3 Humidity 30 2025-09-26 13:38:49"}
{"timestamp": "2025-10-02T04:38:50", "level": "INFO", "msg": "processed 3 rows."}
```


take the sql/insert.sql file and execute it as a query



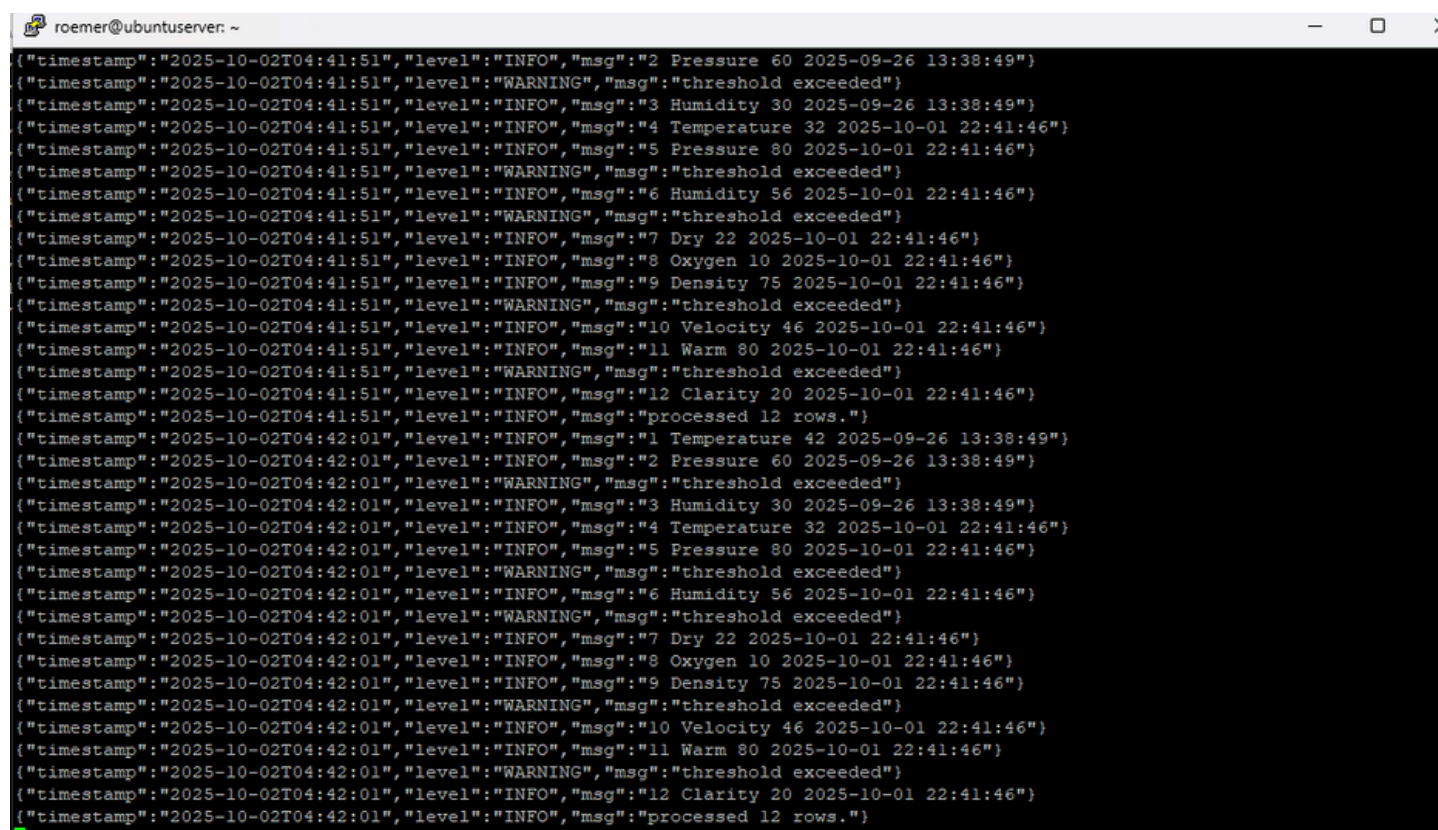
The screenshot shows a SQL IDE window titled 'insert'. The main editor contains a SQL query to insert 10 records into a 'metrics' table. The query is as follows:

```
1 /* inserting 10 new records metrics table */
2 INSERT INTO metrics (name, value) VALUES
3 ('Temperature', 32),
4 ('Pressure', 80),
5 ('Humidity', 56),
6 ('Dry', 22),
7 ('Oxygen', 10),
8 ('Density', 75),
9 ('Velocity', 46),
10 ('Warm', 80),
11 ('Clarity', 20);
12
13
```

Below the editor, the 'Output' pane shows the 'Action Output' for the query. It displays a single row of information:

#	Time	Action	Message
1	22:41:46	INSERT INTO metrics (name, value) VALUES ('Temperature', 32), ('Pressure', 80), ('Humidity', 56), ('Dry', 22), ('Oxygen', 10), ('Density', 75), ('Velocity', 46), ('Warm', 80), ('Clarity', 20);	9 row(s) affected Records: 9 Duplicates: 0 Warnings: 0

new information is added in logs for monitor



The screenshot shows a terminal window with the prompt 'roemer@ubuntu: ~'. It displays a series of log entries in JSON format. The logs show the insertion of 12 rows of data into the 'metrics' table. Each row is logged with a timestamp, level (INFO or WARNING), and message. The messages include the metric name, value, and a timestamp. The logs are as follows:

```
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "INFO",
  "msg": "2 Pressure 60 2025-09-26 13:38:49"
}
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "WARNING",
  "msg": "threshold exceeded"
}
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "INFO",
  "msg": "3 Humidity 30 2025-09-26 13:38:49"
}
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "INFO",
  "msg": "4 Temperature 32 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "INFO",
  "msg": "5 Pressure 80 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "WARNING",
  "msg": "threshold exceeded"
}
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "INFO",
  "msg": "6 Humidity 56 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "WARNING",
  "msg": "threshold exceeded"
}
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "INFO",
  "msg": "7 Dry 22 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "INFO",
  "msg": "8 Oxygen 10 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "INFO",
  "msg": "9 Density 75 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "WARNING",
  "msg": "threshold exceeded"
}
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "INFO",
  "msg": "10 Velocity 46 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "INFO",
  "msg": "11 Warm 80 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "WARNING",
  "msg": "threshold exceeded"
}
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "INFO",
  "msg": "12 Clarity 20 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:41:51",
  "level": "INFO",
  "msg": "processed 12 rows."
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "INFO",
  "msg": "1 Temperature 42 2025-09-26 13:38:49"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "INFO",
  "msg": "2 Pressure 60 2025-09-26 13:38:49"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "WARNING",
  "msg": "threshold exceeded"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "INFO",
  "msg": "3 Humidity 30 2025-09-26 13:38:49"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "INFO",
  "msg": "4 Temperature 32 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "INFO",
  "msg": "5 Pressure 80 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "WARNING",
  "msg": "threshold exceeded"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "INFO",
  "msg": "6 Humidity 56 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "WARNING",
  "msg": "threshold exceeded"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "INFO",
  "msg": "7 Dry 22 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "INFO",
  "msg": "8 Oxygen 10 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "INFO",
  "msg": "9 Density 75 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "WARNING",
  "msg": "threshold exceeded"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "INFO",
  "msg": "10 Velocity 46 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "INFO",
  "msg": "11 Warm 80 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "WARNING",
  "msg": "threshold exceeded"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "INFO",
  "msg": "12 Clarity 20 2025-10-01 22:41:46"
}
{
  "timestamp": "2025-10-02T04:42:01",
  "level": "INFO",
  "msg": "processed 12 rows."
}
```

now the request for status shows other json response



The screenshot shows a terminal window with the prompt 'roemer@ubuntu: ~'. It displays the output of a curl command that requests the status of the application. The output is a JSON object:

```
{
  "alerts": 5,
  "last_check": "2025-10-02T04:48:34Z",
  "rows_processed": 12
}
```

