



AARHUS UNIVERSITET

Microservices and DevOps

Scalable Microservices

Humio

Henrik Bærbak Christensen

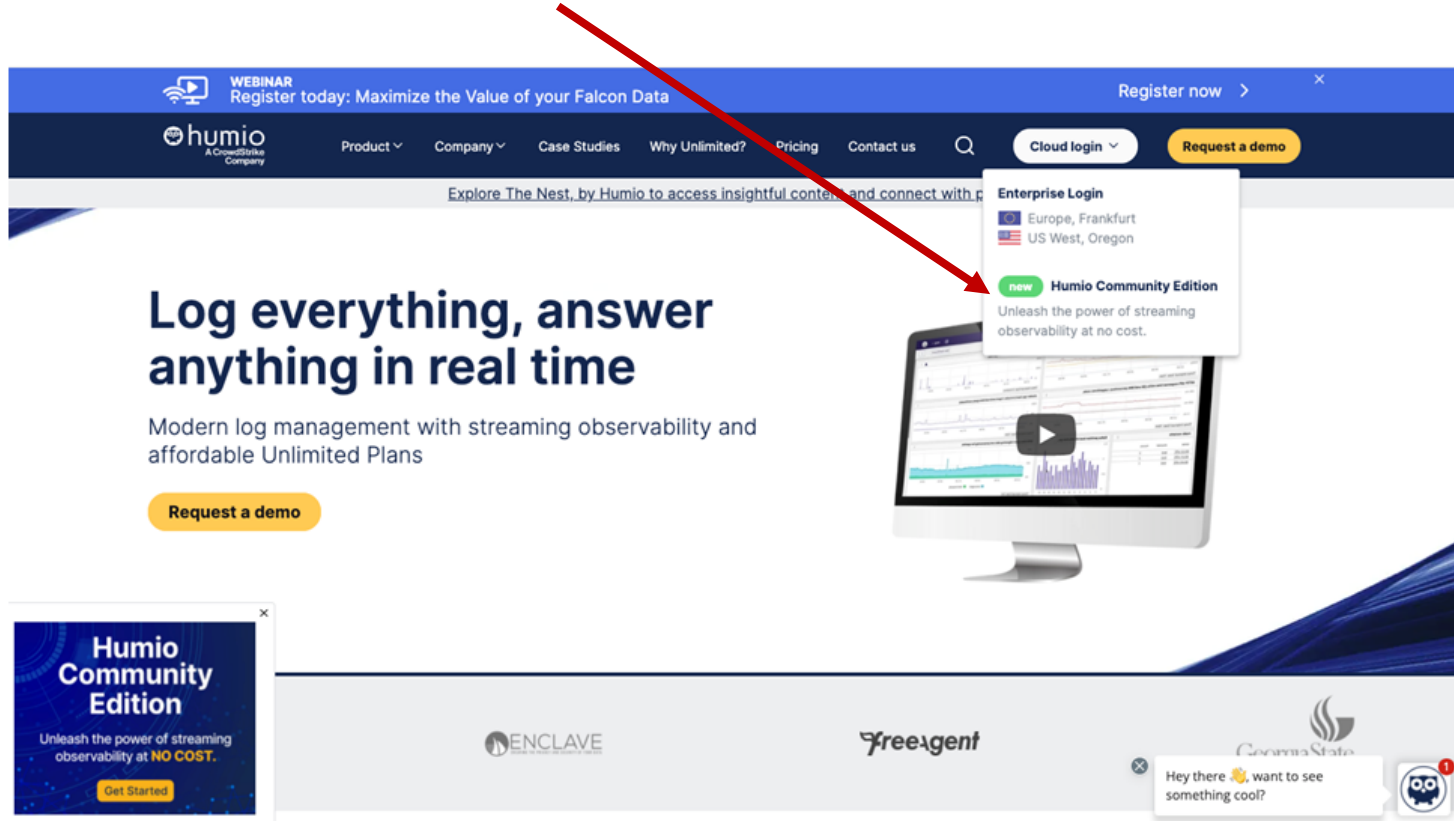
- Humio = Lots of manual machinery!
- When you find it taxing – remember...
- ***This is nothing compared to an ELK stack 😊***
 - I spend vast number of hours on ELK and it was tedious and required a lot of CPU power 😞



AARHUS UNIVERSITET

Setup

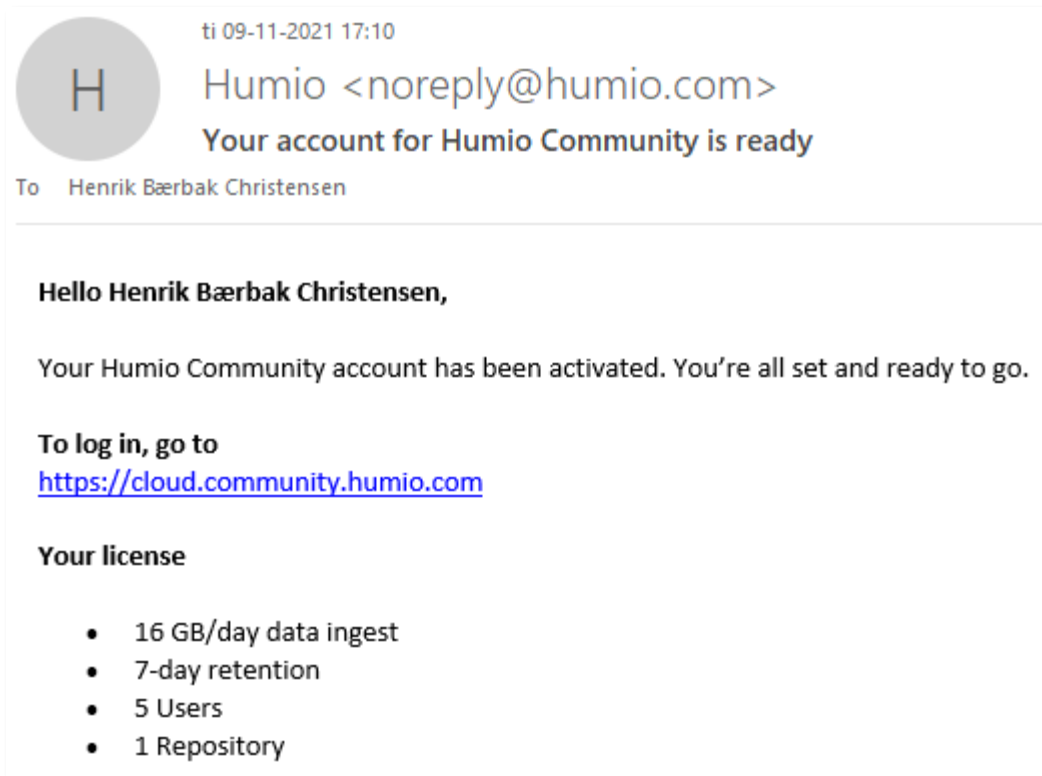
We go for their Community Edition
Cloud Service...



The screenshot shows the Humio website homepage. At the top, there is a blue header with a 'WEBINAR' banner that says 'Register today: Maximize the Value of your Falcon Data' and a 'Register now' link. Below this is a dark blue navigation bar with the Humio logo and links for Product, Company, Case Studies, Why Unlimited?, Pricing, and Contact us. There are also 'Cloud login' and 'Request a demo' buttons. A red arrow points from the top of the page down to the 'Humio Community Edition' button in the 'Enterprise Login' dropdown menu. The main content area features the headline 'Log everything, answer anything in real time' and a sub-headline 'Modern log management with streaming observability and affordable Unlimited Plans'. A 'Request a demo' button is located below the sub-headline. In the bottom left corner, there is a 'Humio Community Edition' badge that says 'Unleash the power of streaming observability at NO COST.' and has a 'Get Started' button. In the bottom right corner, there is a 'Hey there, want to see something cool?' notification bubble and a 'Corona State' logo.

You will need to register...

- And some time (1-2 days – or just a few hours) later, you get access...

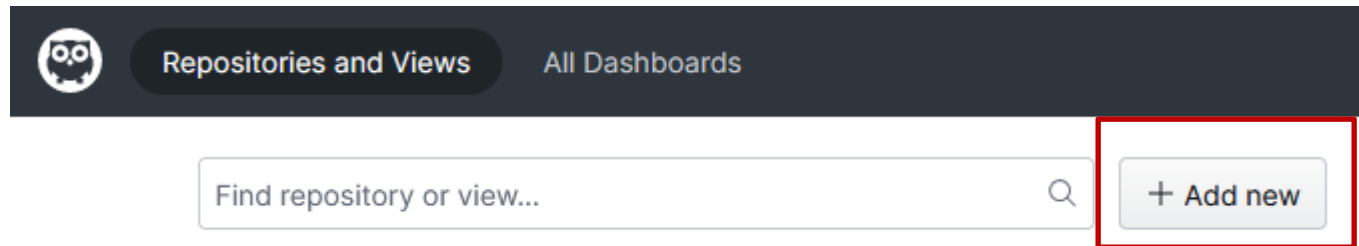


Next Steps – In Overview

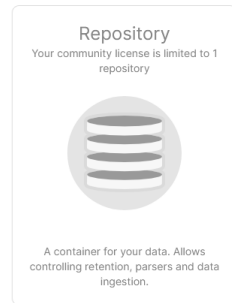
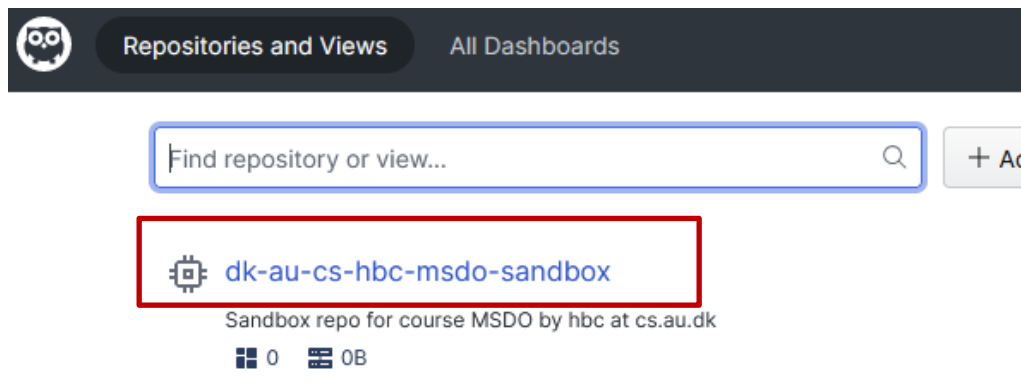
- You need to
 - Define a repository
 - to hold your logging data
 - Define an ingest token
 - That identifies where your log stream's sink is
 - That is – the repository
 - Define a parser
 - So Humio can understand the format of your log messages
 - Make your container(s) ship logs to your repository

Define Repo

- Add repo

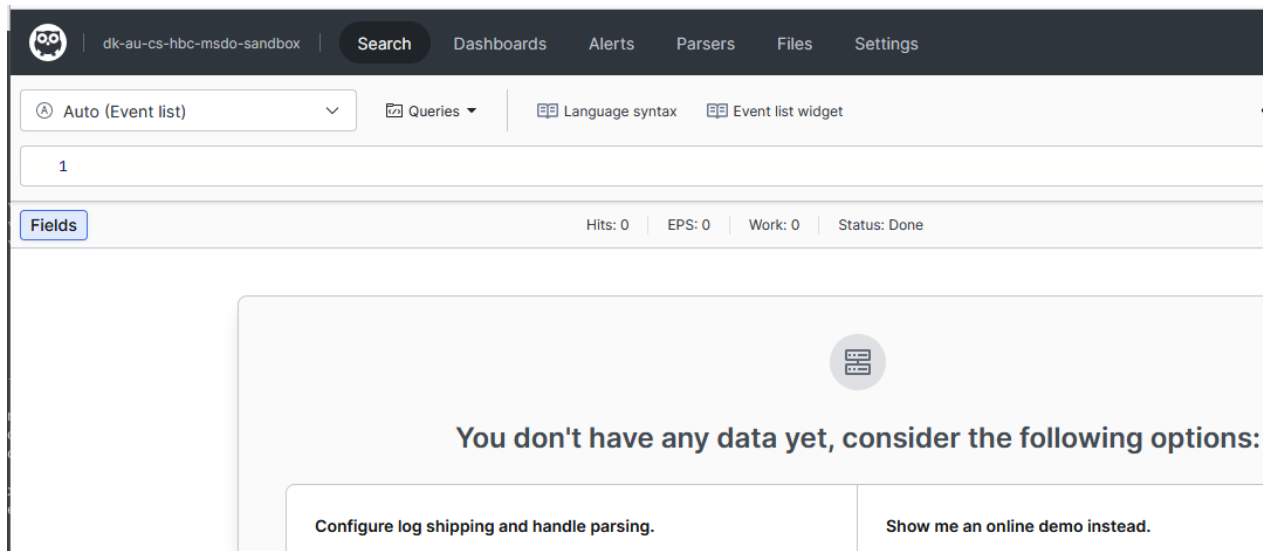


- Hit 'Repository' and define one with a **unique** name



Define Token

- Hit your defined repo to find the *dashboard page*



Do the Tutorial

- Advice to browse over the tutorial
 - One of the super great assets of Humio – is the built-in learning experience !

consider the following options:

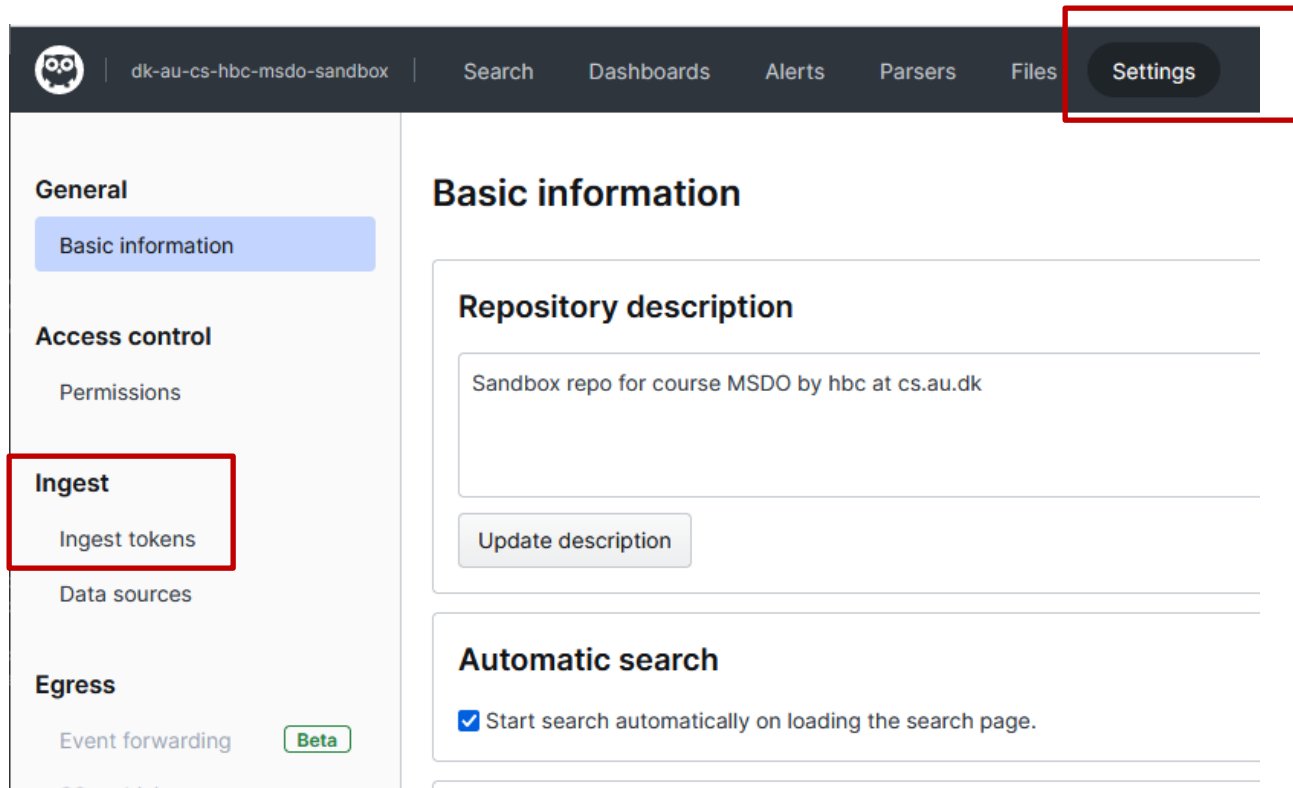
Show me an online demo instead.

If you do not have logs readily available to put into Humio, you can try our **In-App Interactive Tutorial**. The tutorial will generate data for you and that way you don't have to set up log shipping and parsing.

You can always come back to the tutorial by going to Help > Interactive Tutorial in the menu bar.

▶ Start the tutorial

... and now define Ingest Token



dk-au-cs-hbc-msdo-sandbox | Search Dashboards Alerts Parsers Files **Settings**

General

- Basic information

Access control

- Permissions

Ingest

- Ingest tokens**
- Data sources

Egress

- Event forwarding Beta

Basic information

Repository description

Sandbox repo for course MSDO by hbc at cs.au.dk

Update description

Automatic search

☒ Start search automatically on loading the search page.

... ingest token

Ingest tokens

Ingest Tokens are used for authorization when sending data to Humio. Ingest token have limited API access and cannot e.g. be used to read repository settings or execute queries.

[Ingest tokens](#)

Tokens

+ Add token

| Name | Assigned parser | Token |
|---------|-----------------|---|
| default | [None] |   |

Tokens

+ Add token

| Name | Assigned parser | Token |
|---------|-----------------|---|
| default | [None] |   |
| msdo | [None] |   |

Use 'eye' to get the token



Logging to Humio

- Now, the next (big) challenge is
 - Telling Docker to forward logs to Humio
 - Make Humio understand the weird 'Henrik Bærbak' logging format
- Understand what we can do with Humio...

Docker Forwarding

- Fortunately, Humio support in Docker is already built-in
- Just use the 'splunk' logging driver of Docker
 - Tell Docker to use splunk, not the default logging driver
 - Tell Docker which machine to ship the logs to
 - Tell Docker the proper credentials (INGEST_TOKEN)

Ingest Token

- The token must be provided for any docker run. One way:
 - export INGEST_TOKEN=(paste token here)
- Now, start your daemon with the proper settings
 - Set driver
 - Set endpoint URL
 - Humio's endpoint
 - Set ingest token
 - Set format to 'raw'
 - Makes parsing easier...

```
export HUMIO_URL=https://cloud.community.humio.com
```

```
export INGEST_TOKEN=f2c2c
```

```
docker run -d -p 7777:7777 --name daemon \  
  --log-driver=splunk \  
  --log-opt splunk-url=$HUMIO_URL \  
  --log-opt splunk-token=$INGEST_TOKEN \  
  --log-opt splunk-format=raw \  
  henrikbaerbak/private:cave-jar
```



AARHUS UNIVERSITET

Let's see some action

- ... these steps
 - Build an image

```
docker build -f Dockerfile-multistage -t henrikbaerbak/private:cave-humio .
```

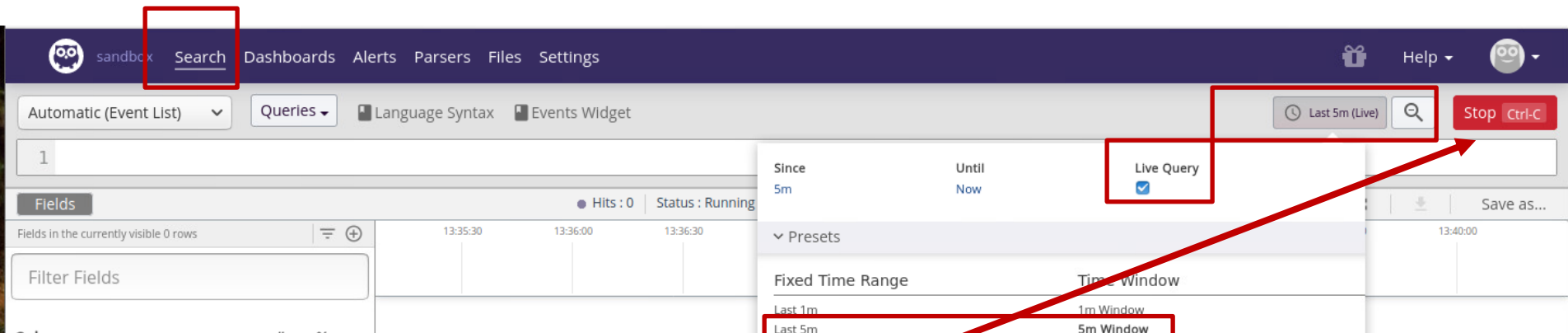
- Set the Humio environment

```
csdev@m1:~/proj/cave$ export HUMIO_URL=https://cloud.community.humio.com
csdev@m1:~/proj/cave$ export INGEST_TOKEN=f2c2c007-54f0-4212-8c27-7b2651d11111
```

- Run the daemon

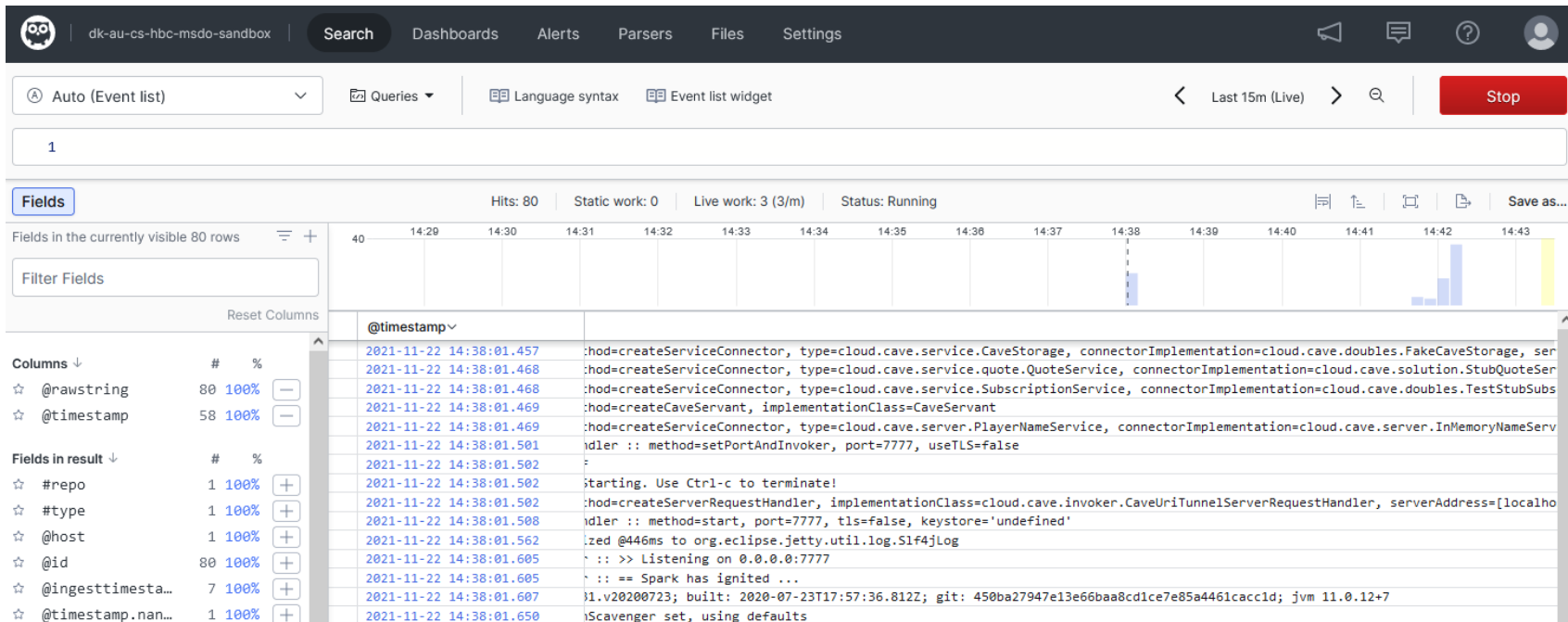
```
docker run -d -p 7777:7777 --name daemon \
  --log-driver=splunk \
  --log-opt splunk-url=$HUMIO_URL \
  --log-opt splunk-token=$INGEST_TOKEN \
  --log-opt splunk-format=raw \
  henrikbaerbak/private:cave-humio
```


- You should see some log messages coming in...
 - ... if you set the search for 5 minutes / live



The screenshot shows the Kibana Search interface. The 'Search' tab is selected in the top navigation bar. The search bar contains the text '1'. Below the search bar, the 'Fields' section is visible. The 'Live Query' section is expanded, showing the 'Live Query' checkbox checked. The 'Presets' section is also visible, with 'Last 5m' and '5m Window' selected. A red arrow points from the 'run' button (labeled 'Stop Ctrl-C') to the 'Live Query' checkbox.

- And hit the 'run' button



But – No Parsing

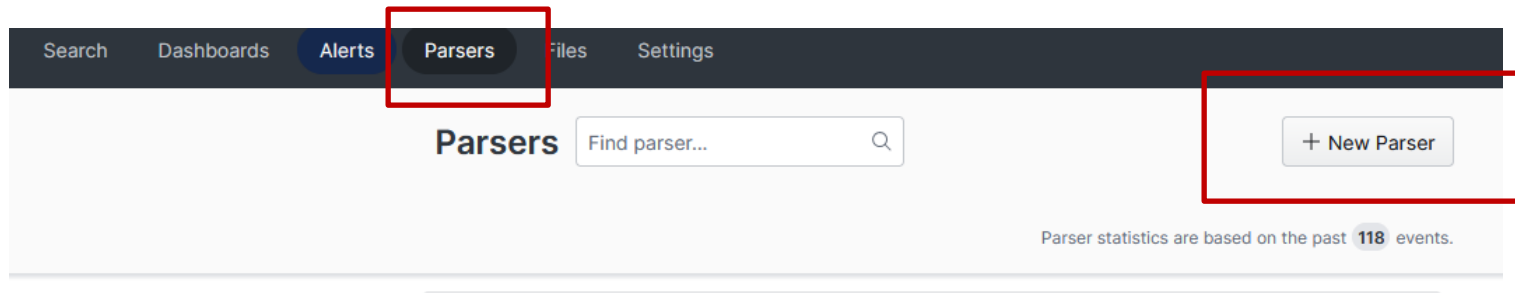
- Humio understands most common logging formats
 - Accesslog, syslog, json, key-value
- However, our Log4J property file defines its own
 - Why? Well...

```
log4j.appender.DockerConsole.layout=org.apache.log4j.PatternLayout
log4j.appender.DockerConsole.layout.ConversionPattern=%d{yyyy-MM-dd'T'HH:mm:ss.SSSXXX} [%p] %c :: %m%n
```

- What is the problem?
 - We cannot make queries as each log is just a raw string ☹️

A Parser

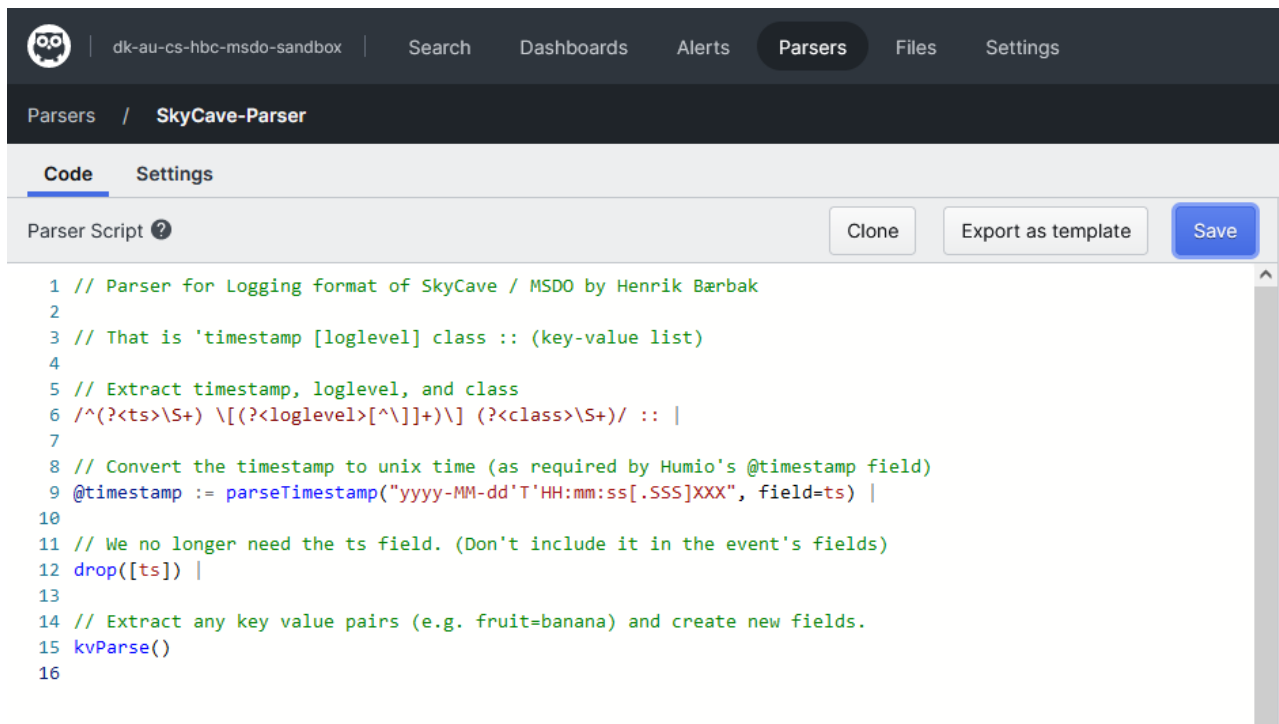
- You will need to define a new parser



A Parser

- And as my format is (mostly) a key-value based one, enter:

- *Find the file on the weekplan !*



The screenshot shows the Humio web interface. The top navigation bar includes a logo, the text 'dk-au-cs-hbc-msdo-sandbox', and links for Search, Dashboards, Alerts, Parsers, Files, and Settings. The 'Parsers' tab is selected. Below the navigation bar, the breadcrumb 'Parsers / SkyCave-Parser' is visible. The main content area has two tabs: 'Code' (selected) and 'Settings'. Under the 'Code' tab, there is a 'Parser Script' section with a help icon. To the right of the script are three buttons: 'Clone', 'Export as template', and 'Save'. The script itself is a Humio query in a code editor, showing comments and code for parsing SkyCave logs. The script includes a regular expression to extract timestamp, loglevel, and class, and uses Humio's built-in functions like `parseTimestamp` and `kvParse`.

```
1 // Parser for Logging format of SkyCave / MSDO by Henrik Bærbak
2
3 // That is 'timestamp [loglevel] class :: (key-value list)
4
5 // Extract timestamp, loglevel, and class
6 /^(?<ts>\S+) \[(?<loglevel>[^\]]+\) \] (?<class>\S+)/ :: |
7
8 // Convert the timestamp to unix time (as required by Humio's @timestamp field)
9 @timestamp := parseTimestamp("yyyy-MM-dd'T'HH:mm:ss[.SSS]XXX", field=ts) |
10
11 // We no longer need the ts field. (Don't include it in the event's fields)
12 drop([ts]) |
13
14 // Extract any key value pairs (e.g. fruit=banana) and create new fields.
15 kvParse()
16
```

And associate it...

- With your token

Ingest tokens

Ingest Tokens are used for authorization when sending data to Humio. Ingest token have limited API access and cannot e.g. be used to read repository settings or execute queries.

 [Ingest tokens](#)

Tokens

+ Add token

| Name | Assigned parser | Token | |
|---------|-----------------|---|---|
| default | [None] |  |  |
| msdo | SkyCave-Parser |  |  |

Unfortunately!

- The most interesting log message is about the daemon accepting and replying to the 'cmd' requests
 - The log message stems from the FRDS.Broker library, which unfortunately has the interesting stuff encoded in a json object ☹️

```
2021-11-10T11:06:28.281+01:00 [INFO] frds.broker.ipc.http.UriTunnelServerRequestHandler :: method=POST, context=request request={"operationName":"playe  
r-get-short-room-description","payload":[],"objectId":"user-001##token#0","versionIdentity":5}  
2021-11-10T11:06:28.282+01:00 [INFO] frds.broker.ipc.http.UriTunnelServerRequestHandler :: method=handleRequest, context=reply, reply={"payload":"\nYou  
are in open forest, with a deep valley to one side.\n","statusCode":200,"versionIdentity":5}, responseTime_ms=1
```

- I have *not* been able to create a parser that can handle that format (would love if someone cracked that nut!)
 - Even if Humio states I can: “kvParse() | parseJson(field=reply)”
 - I just get ‘reply=payload’ ???

So, to monitor commands...

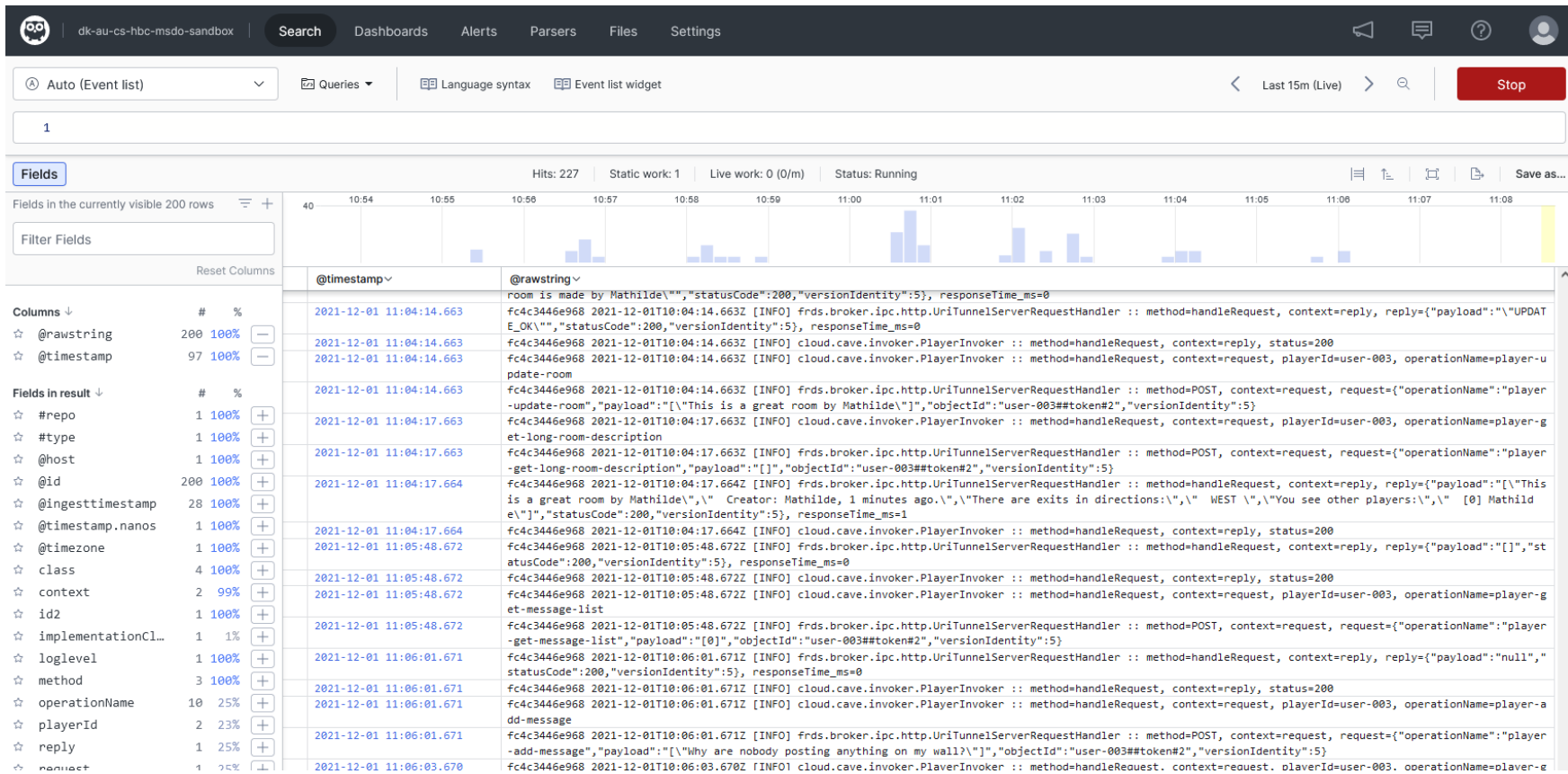
- We have to augment the CaveInvoker's and PlayerInvoker's *handleRequest()*
 - Find updated code on the weekplan.

```
// Added for Humio logging
```

```
logger.info("method=handleRequest, context=request, playerId={}, operationName={}", playerId, operationName);
```

```
// Added for Humio logging
```

```
logger.info("method=handleRequest, context=reply, status={}", reply.getStatusCode());
```

Meta: gradle cmd -Pcpf=malkia.cpf

Selecting Columns

Fields

Fields in the currently visible 200 rows

Filter Fields

Reset Columns

Columns ↓

| | # | % | |
|-------------------|----|------|---|
| ☆ @timestamp | 97 | 100% | — |
| ☆ operationName | 10 | 25% | — |
| ☆ playerId | 2 | 23% | — |
| ☆ responseTime_ms | 7 | 25% | — |
| ☆ status | 2 | 25% | — |

Fields in result ↓

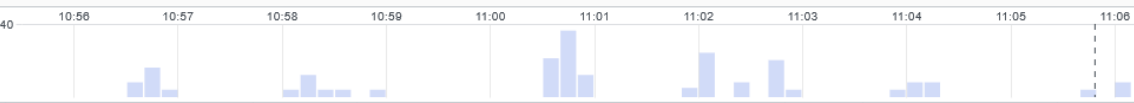
| | # | % | |
|-----------------------|-----|------|---|
| ☆ #repo | 1 | 100% | + |
| ☆ #type | 1 | 100% | + |
| ☆ @host | 1 | 100% | + |
| ☆ @id | 200 | 100% | + |
| ☆ @ingesttimestamp | 28 | 100% | + |
| ☆ @rawstring | 200 | 100% | + |
| ☆ @timestamp.nanos | 1 | 100% | + |
| ☆ @timezone | 1 | 100% | + |
| ☆ class | 4 | 100% | + |
| ☆ context | 2 | 99% | + |
| ☆ id2 | 1 | 100% | + |
| ☆ implementationCl... | 1 | 1% | + |
| ☆ loglevel | 1 | 100% | + |
| ☆ method | 3 | 100% | + |
| ☆ reply | 1 | 25% | + |

Hits: 218

Static work: 1

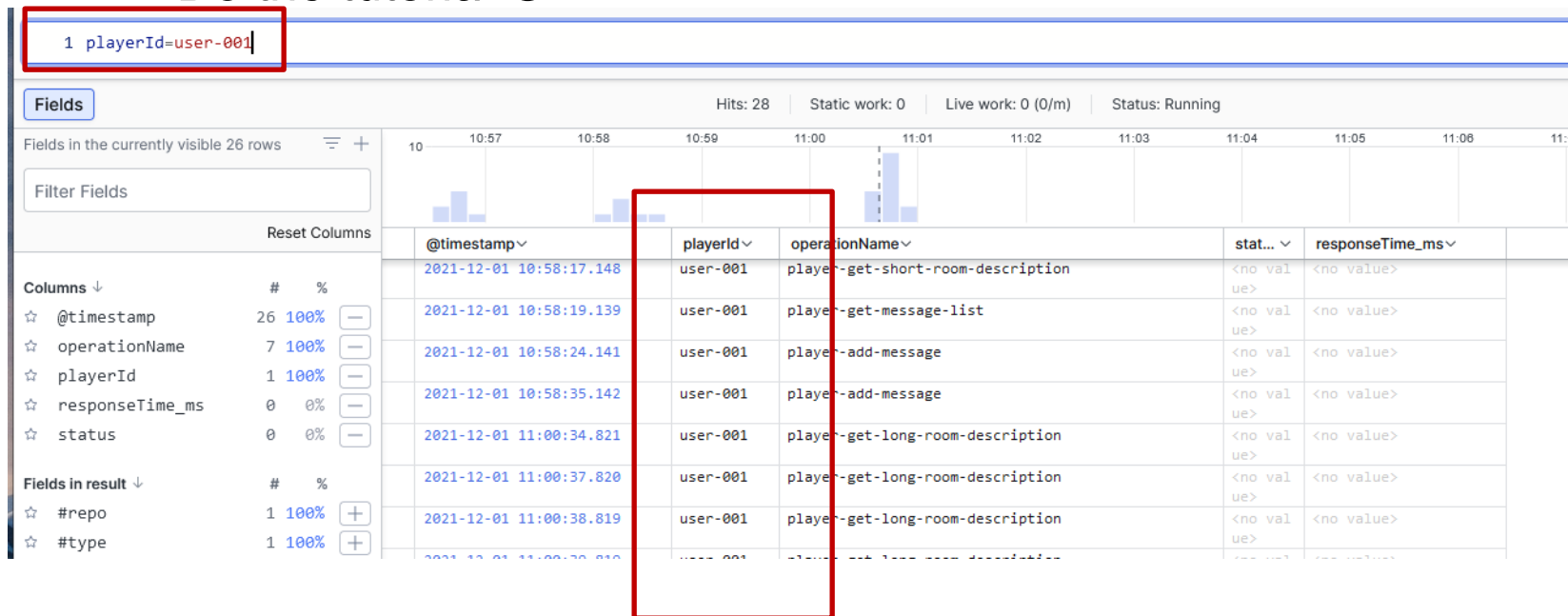
Live work: 3 (2/m)

Status: Running

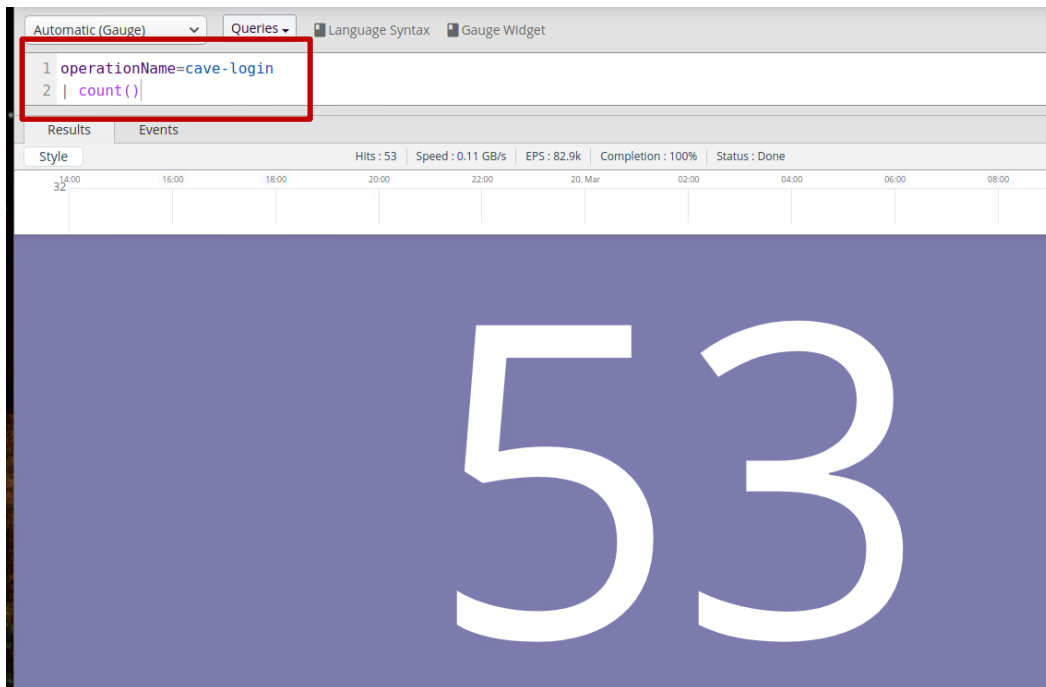


| @timestamp▼ | playerId▼ | operationName▼ | stat...▼ | responseTime_ms▼ |
|-------------------------|------------|----------------------------------|-------------|------------------|
| 2021-12-01 11:04:03.671 | <no value> | <no value> | <no val ue> | 0 |
| 2021-12-01 11:04:14.663 | <no value> | <no value> | <no val ue> | 0 |
| 2021-12-01 11:04:14.663 | <no value> | <no value> | 200 | <no value> |
| 2021-12-01 11:04:14.663 | user-003 | player-update-room | <no val ue> | <no value> |
| 2021-12-01 11:04:14.663 | <no value> | <no value> | <no val ue> | <no value> |
| 2021-12-01 11:04:17.663 | user-003 | player-get-long-room-description | <no val ue> | <no value> |
| 2021-12-01 11:04:17.663 | <no value> | <no value> | <no val ue> | <no value> |
| 2021-12-01 11:04:17.664 | <no value> | <no value> | <no val ue> | 1 |
| 2021-12-01 11:04:17.664 | <no value> | <no value> | 200 | <no value> |
| 2021-12-01 11:05:48.672 | <no value> | <no value> | <no val ue> | 0 |
| 2021-12-01 11:05:48.672 | <no value> | <no value> | 200 | <no value> |
| 2021-12-01 11:05:48.672 | user-003 | player-get-message-list | <no val ue> | <no value> |
| 2021-12-01 11:05:48.672 | <no value> | <no value> | <no val ue> | <no value> |
| 2021-12-01 11:06:01.671 | <no value> | <no value> | <no val ue> | 0 |
| 2021-12-01 11:06:01.671 | <no value> | <no value> | 200 | <no value> |
| 2021-12-01 11:06:01.671 | user-003 | player-add-message | <no val ue> | <no value> |
| 2021-12-01 11:06:01.671 | <no value> | <no value> | <no val ue> | <no value> |
| 2021-12-01 11:06:03.670 | user-003 | player-get-message-list | <no val ue> | <no value> |

- You can use a query language
 - Do the tutorial 😊

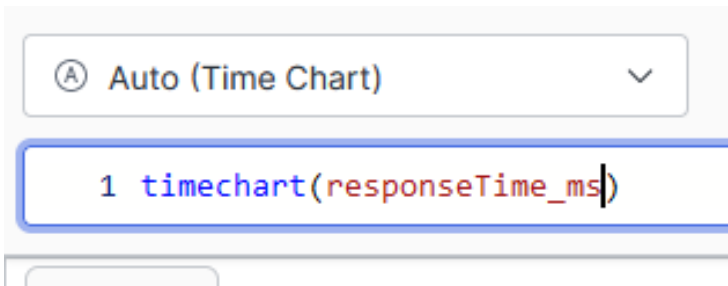


Semi SQL like 😊

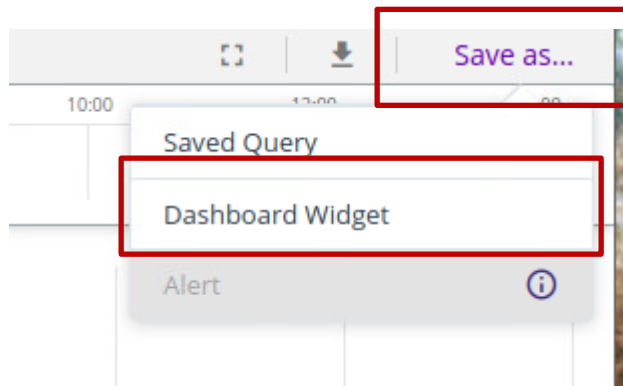


Dashboards

- Perform a search



- Hit 'Save as...'
 - And add to a dashboard



Dashboard Example

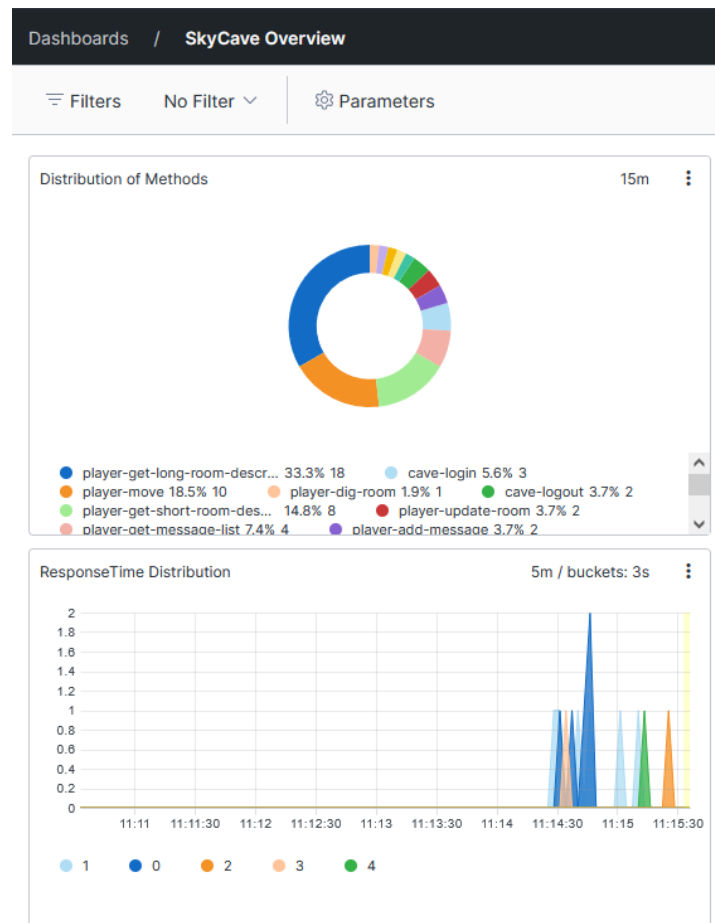
- Here I have added two widgets

- Distribution of Methods

- Query: `operationName=* | groupBy(operationName)`
- As Pie Chart

- Response time distribution

- As TimeChart





AARHUS UNIVERSITET

Swarm

More fiddling 😞

The IngestToken

- Easy to use an environment variable on a single node
- For swarm it is a bit more tricky:

environment

Add environment variables. You can use either an array or a dictionary. Any boolean values (true, false, yes, no) need to be enclosed in quotes to ensure they are not converted to True or False by the YML parser.

Environment variables with only a key **are resolved to their values on the machine Compose is running on**, which can be helpful for secret or host-specific values.

- Alas, you have to set it on *each machine* in swarm 😞

To make Environment variables persistent you need to define those variables in the bash configuration files. In most Linux distributions when you start a new session, environment variables are read from the following files:

- **/etc/environment** - Use this file to set up system-wide environment variables. Variables in this file are set in the following format:

```
$ F00=bar  
$ VAR_TEST="Test Var"
```

Fiddling 😞


And the compose-file

- ... and then you have to adopt your compose file to set the same properties as for the 'docker run' but now in YAML format...

```
docker run -d -p 7777:7777 \  
  --log-driver=splunk \  
  --log-opt splunk-url=http://localhost:8080 \  
  --log-opt splunk-token=$INGEST_TOKEN \  
  --log-opt splunk-format=raw \  
  henrikbaerbak/private:cave-jar
```

- Jobs done...

- Alternative:
 - Expose your ingest token directly in the compose file. Hm hm...



```
logging:  
  driver: splunk  
  options:  
    splunk-url: "http://ml-
```

And more 😊

Summary

- Log aggregation is a powerful tool
- Humio has a (comparable) gentle learning curve
- Quite a lot of moving parts still 😊...