



Microservices and DevOps

Scalable Microservices

Humio

Henrik Bærbak Christensen

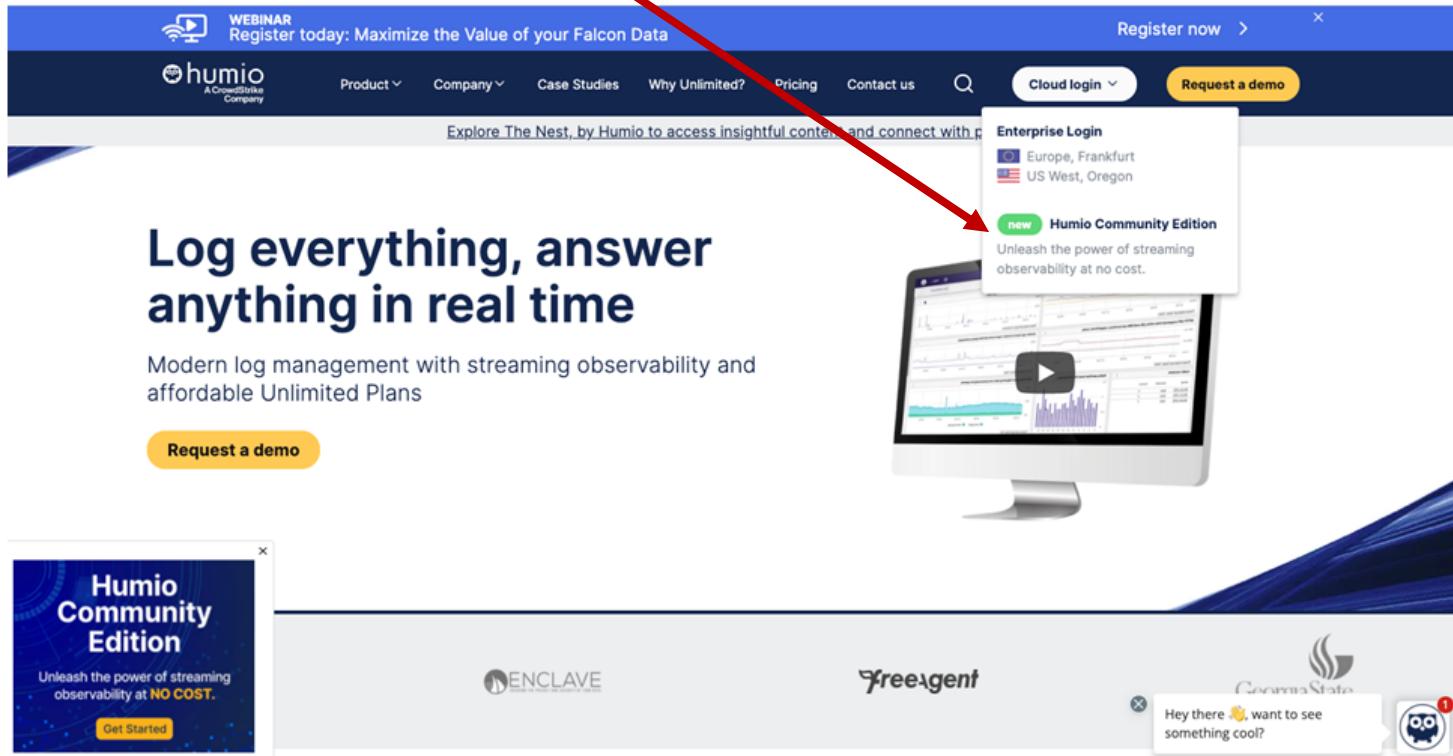
- Humio = Lots of manual machinery!
- When you find it taxing – remember...
- ***This is nothing compared to an ELK stack 😊***
 - I spend vast number of hours on ELK and it was tedious and required a lot of CPU power 😞



Setup

We go for their Community Edition
Cloud Service...

Just Click...



WEBINAR Register today: Maximize the Value of your Falcon Data

humio A CrowdStrike Company

Product Company Case Studies Why Unlimited? Pricing Contact us

Cloud login Request a demo

Explore The Nest, by Humio to access insightful content and connect with people.

Log everything, answer anything in real time

Modern log management with streaming observability and affordable Unlimited Plans

Request a demo

Humio Community Edition

Unleash the power of streaming observability at **NO COST**.

Get Started

ENCLAVE

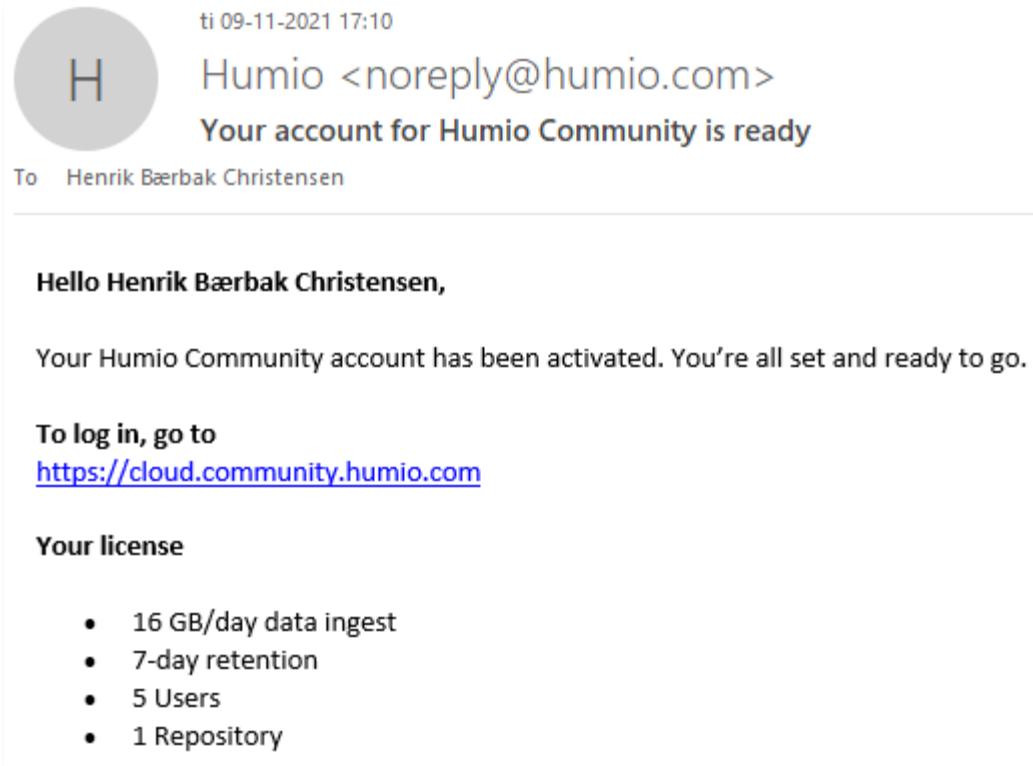
freeagent

GammaState

Hey there, want to see something cool?

You will need to register...

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



ti 09-11-2021 17:10

Humio <noreply@humio.com>

Your account for Humio Community is ready

To Henrik Bærbak Christensen

Hello Henrik Bærbak Christensen,

Your Humio Community account has been activated. You're all set and ready to go.

To log in, go to
<https://cloud.community.humio.com>

Your license

- 16 GB/day data ingest
- 7-day retention
- 5 Users
- 1 Repository

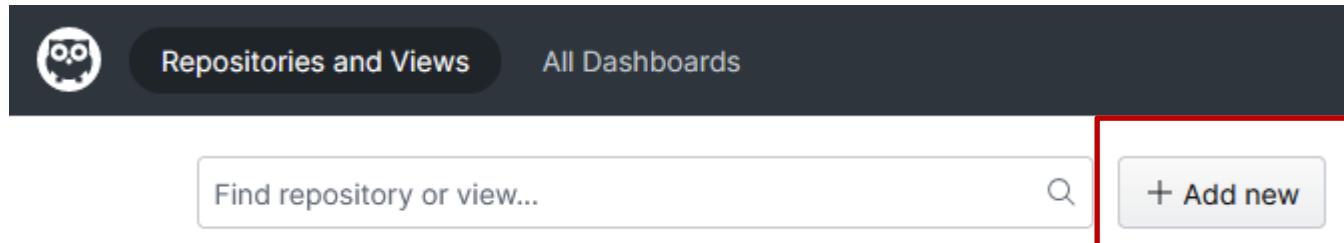


Next Steps – In Overview

- 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

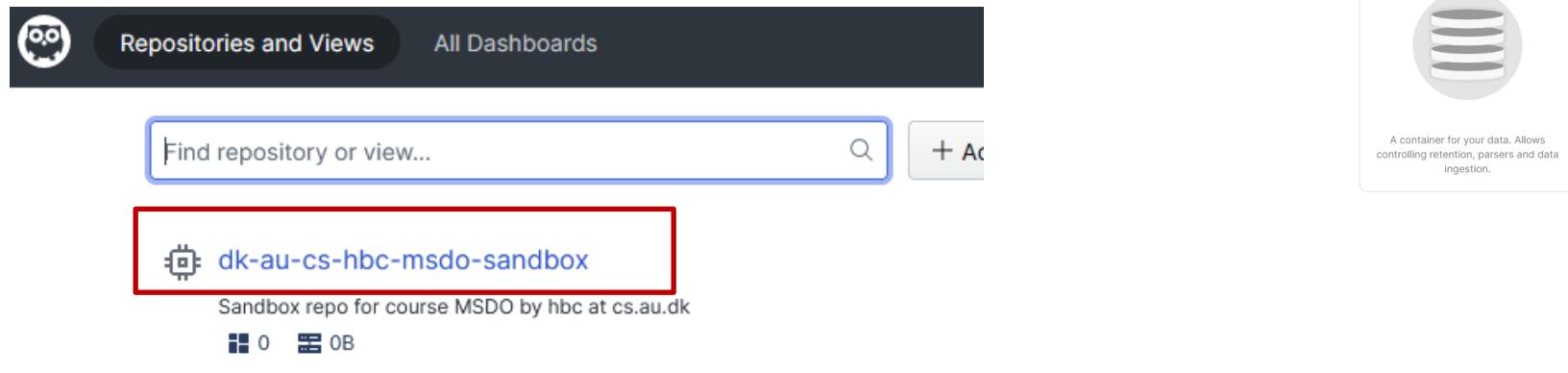
Define Repo

- Add repo



The screenshot shows a dark-themed interface for managing repositories. At the top, there are tabs for 'Repositories and Views' (which is active and highlighted in a dark grey box) and 'All Dashboards'. Below the tabs is a search bar with the placeholder 'Find repository or view...' and a magnifying glass icon. To the right of the search bar is a button labeled '+ Add new' with a red box drawn around it. The background is dark, and the overall layout is clean and modern.

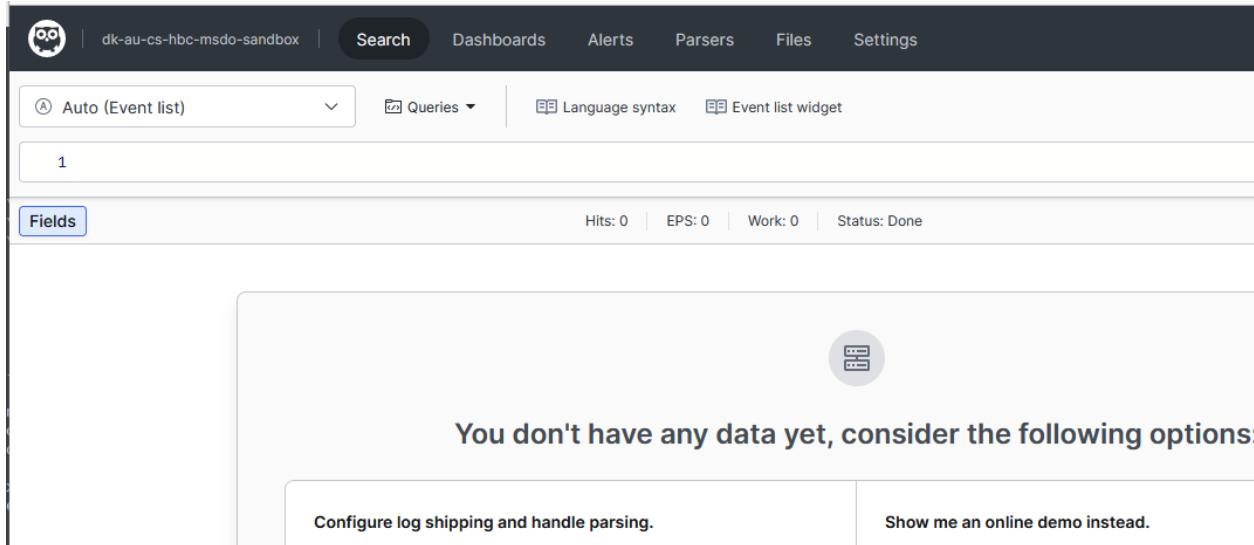
- Hit 'Repository' and define one with a **unique** name



The screenshot shows the same interface after a repository has been added. The search bar is now highlighted with a blue border. Below the search bar, a newly created repository is listed: 'dk-au-cs-hbc-msdo-sandbox'. This repository is highlighted with a red box. The description below the repository name is: 'Sandbox repo for course MSDO by hbc at cs.au.dk'. To the right of the repository list, there is a sidebar with the title 'Repository' and a sub-note: 'Your community license is limited to 1 repository'. Below this, there is an icon of a server and a text box stating: 'A container for your data. Allows controlling retention, parsers and data ingestion.' The overall layout remains consistent with the first screenshot.

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:

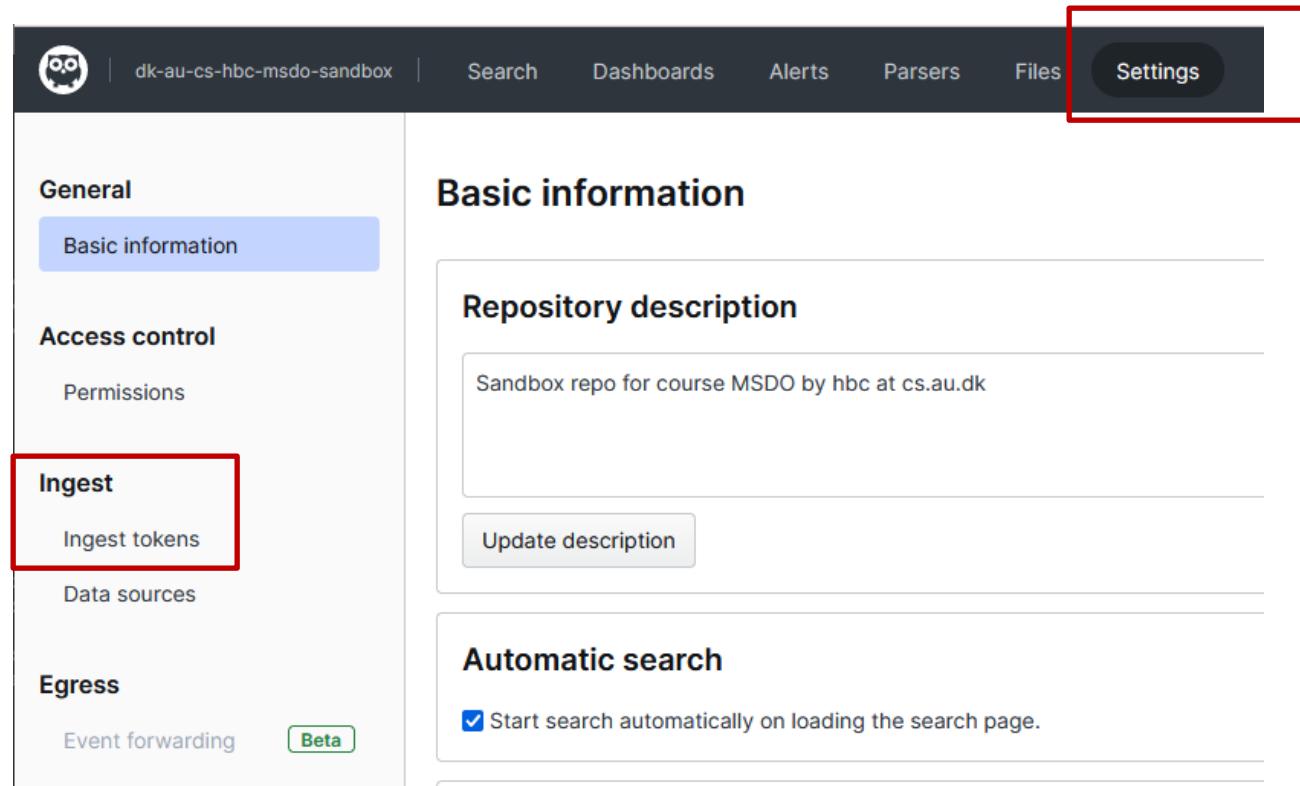
Show me an online demo instead.

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

File sources

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
```



Let's see some action

- ... these steps
 - Build an image

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

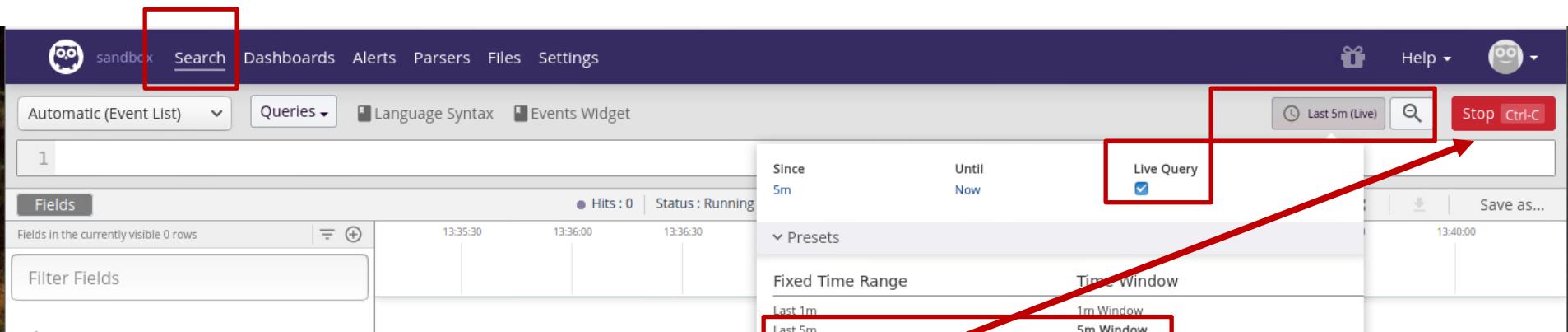
- Set the Humio environment

```
csdev@m1:~/proj/cave$      export HUMIO_URL=https://cloud.community.humio.com
csdev@m1:~/proj/cave$      export INGEST_TOKEN=f2c2c...
```

- Run the daemon

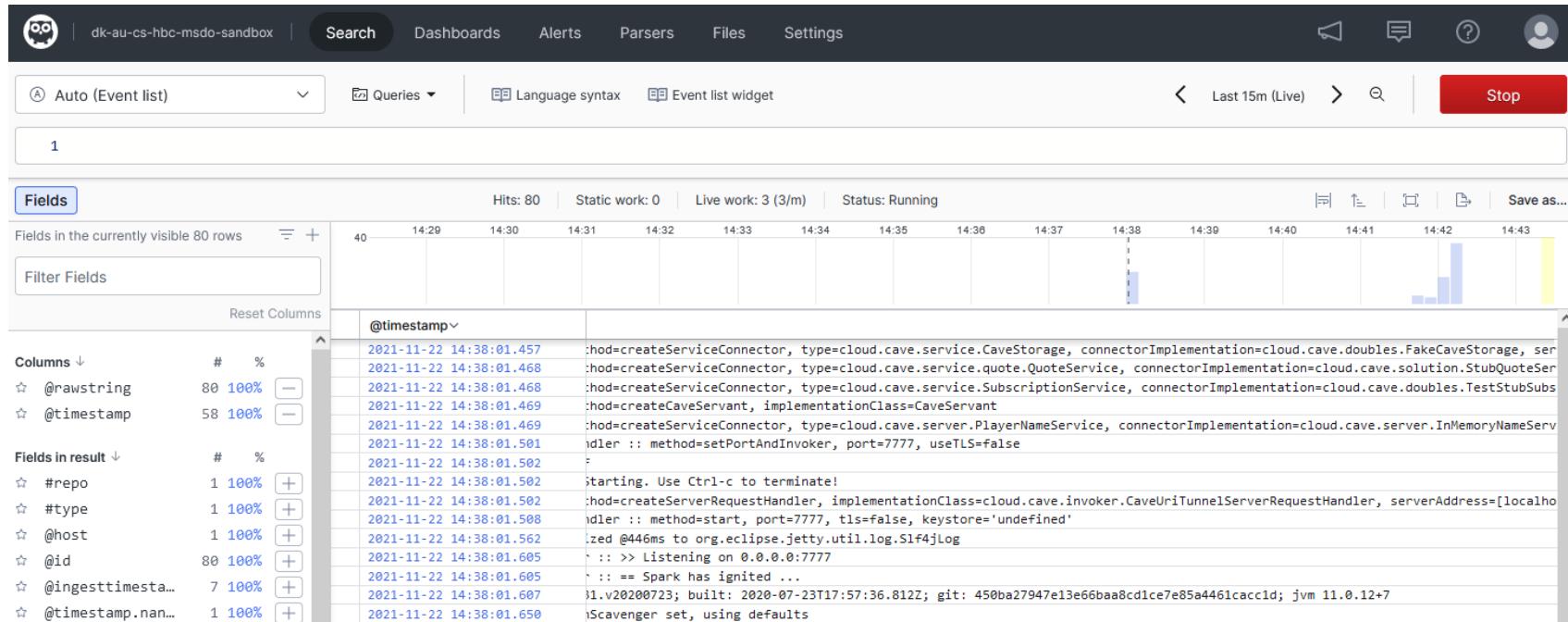
```
docker run -d -p 7777:7777 --name daemon \
  --log-driver=splunk \
  --log-opt splunk-url=$HUMIO_URL \
  --log-opt splunk-token=$INGEST_TOKEN \
  --log-opt splunk-format=raw \
  henrikbaerbak/private:cave-humio
```

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



- And hit the 'run' button

Example



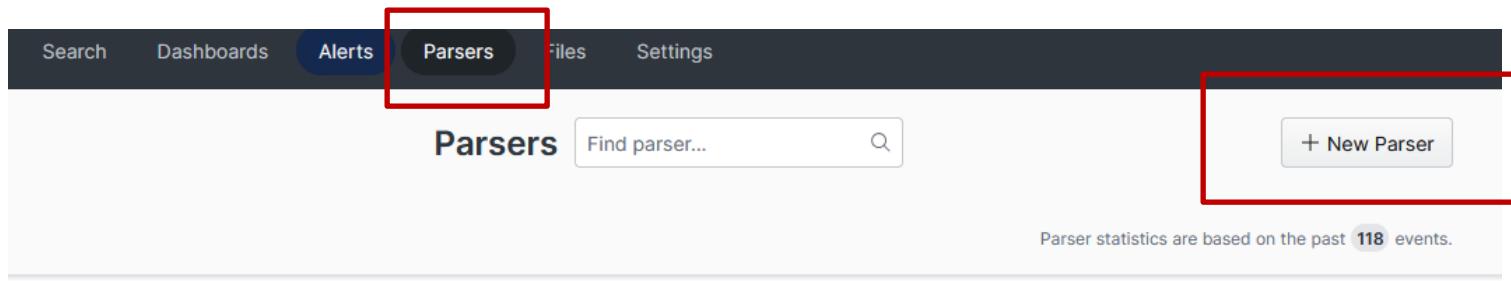
But – No Parsing

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

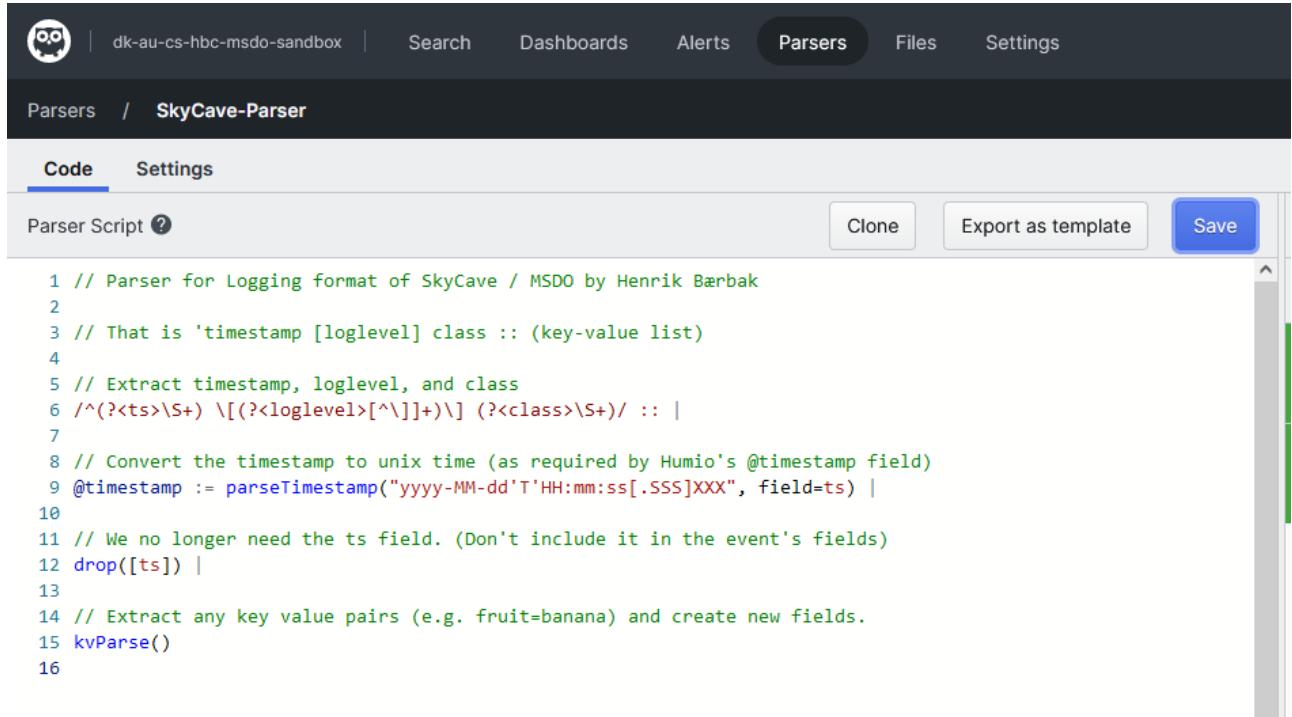
```
log4j.appender.DockerConsole.layout=org.apache.log4j.PatternLayout
log4j.appender.DockerConsole.layout.ConversionPattern=%d{yyyy-MM-dd'T'HH:mm:ss.SSSXXX} [%p] %c :: %m%n
```

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

- You will need to define a new parser



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



The screenshot shows a web-based interface for managing parsers. The top navigation bar includes 'dk-au-cs-hbc-msdo-sandbox', 'Search', 'Dashboards', 'Alerts', 'Parsers' (which is the active tab), 'Files', and 'Settings'. Below the navigation is a breadcrumb trail 'Parsers / SkyCave-Parser'. The main content area has two tabs: 'Code' (selected) and 'Settings'. The 'Parser Script' code is displayed in a code editor with syntax highlighting. The code is a Logstash configuration script for parsing SkyCave logs:

```
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
```

Buttons for 'Clone', 'Export as template', and 'Save' are visible on the right side of the code editor.

- 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": "\\"You
are in open forest, with a deep valley to one side.\\"", "statusCode": 200, "versionIdentity": 5}, responseTime_ms=1
```

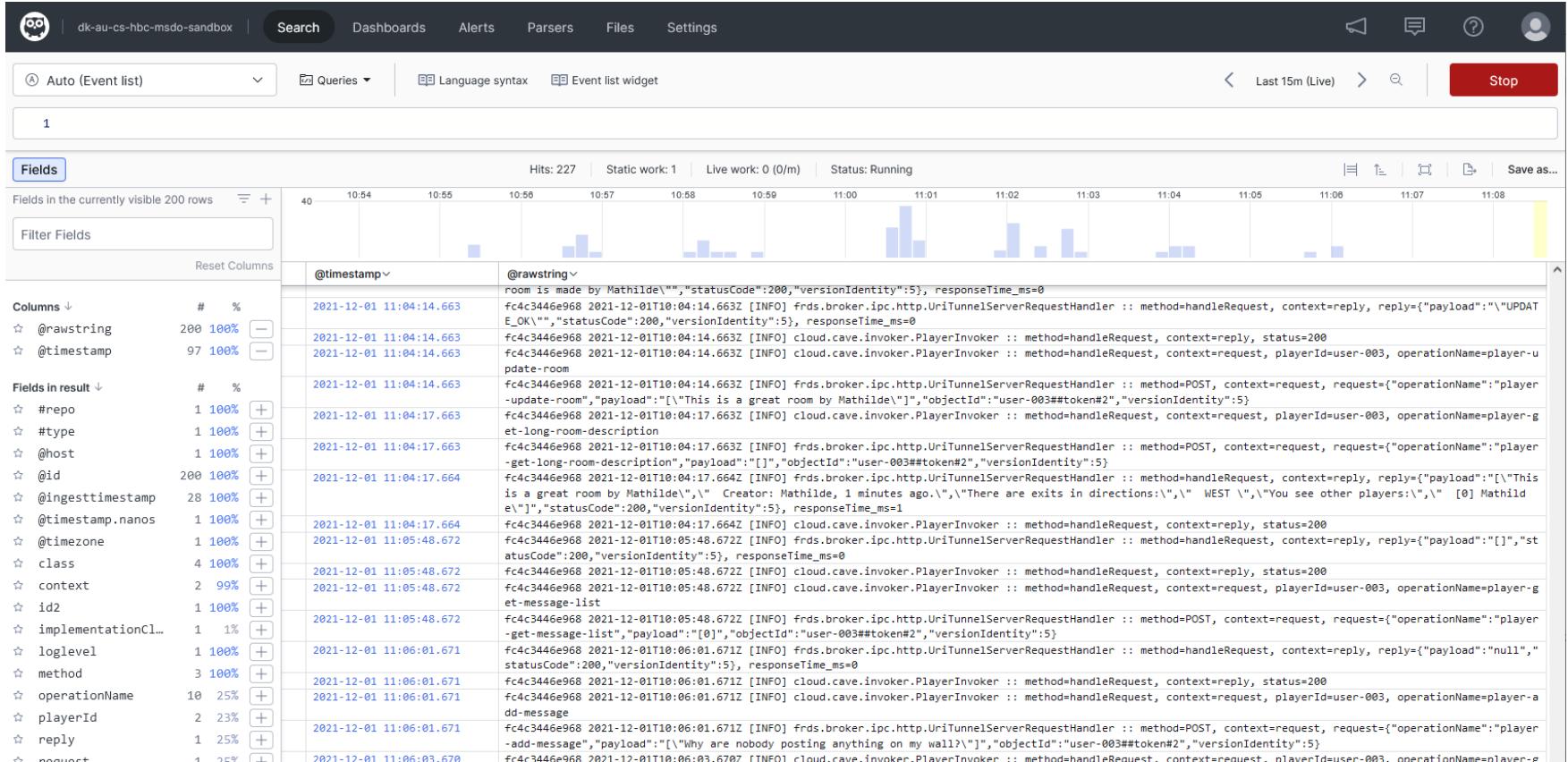
- I have *not* been able to create a parser that can handle that format (would love if someone cracked that nut!)
 - Even if Humio states I can: “kvParse() | parseJson(field=reply)”
 - I just get ‘reply=payload’ ???

So, to monitor commands...

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

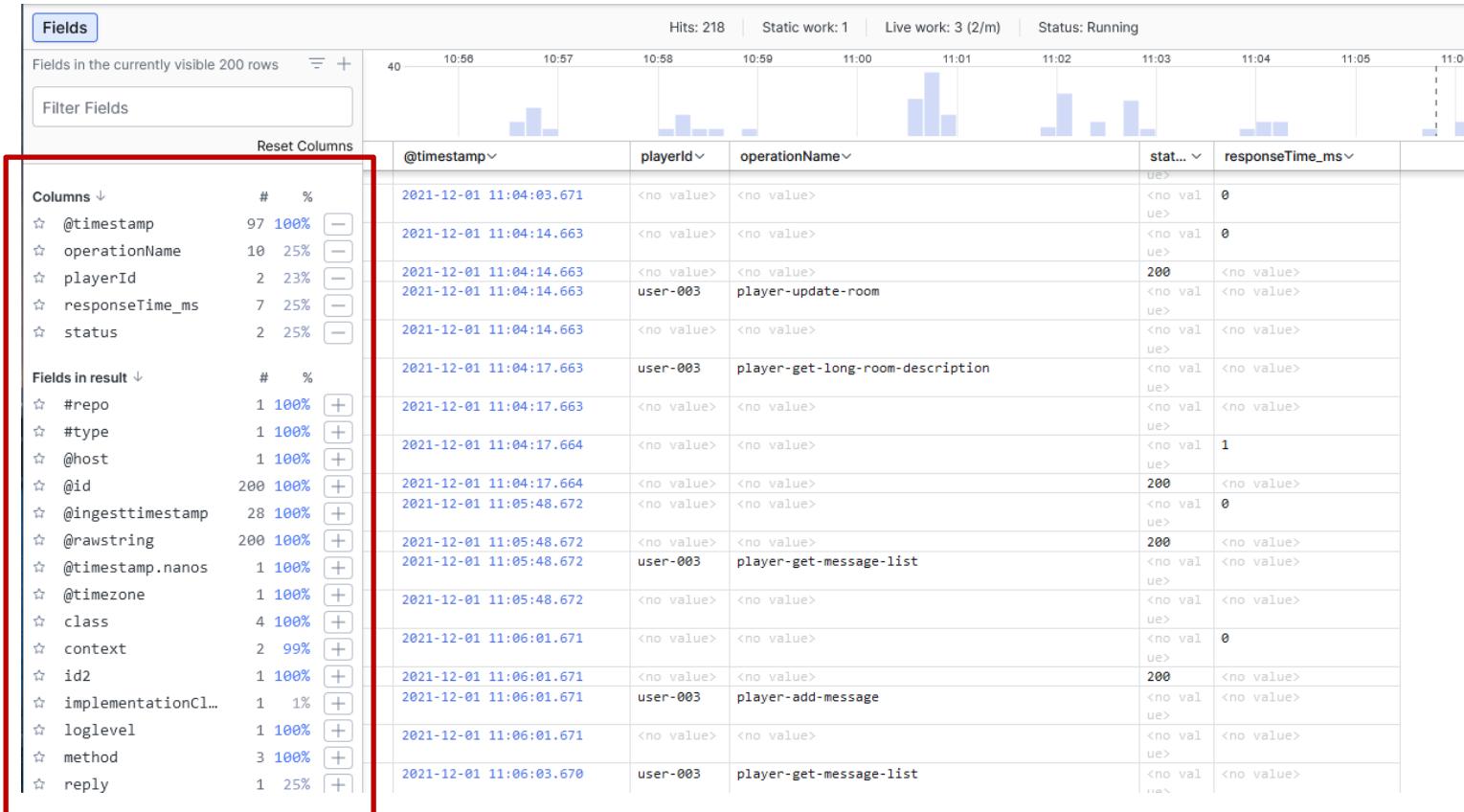
```
// Added for Humio logging
logger.info("method=handleRequest, context=request, playerId={}, operationName={}", playerId, operationName);
```

```
// Added for Humio logging
logger.info("method=handleRequest, context=reply, status={}", reply.getStatusCode());
```

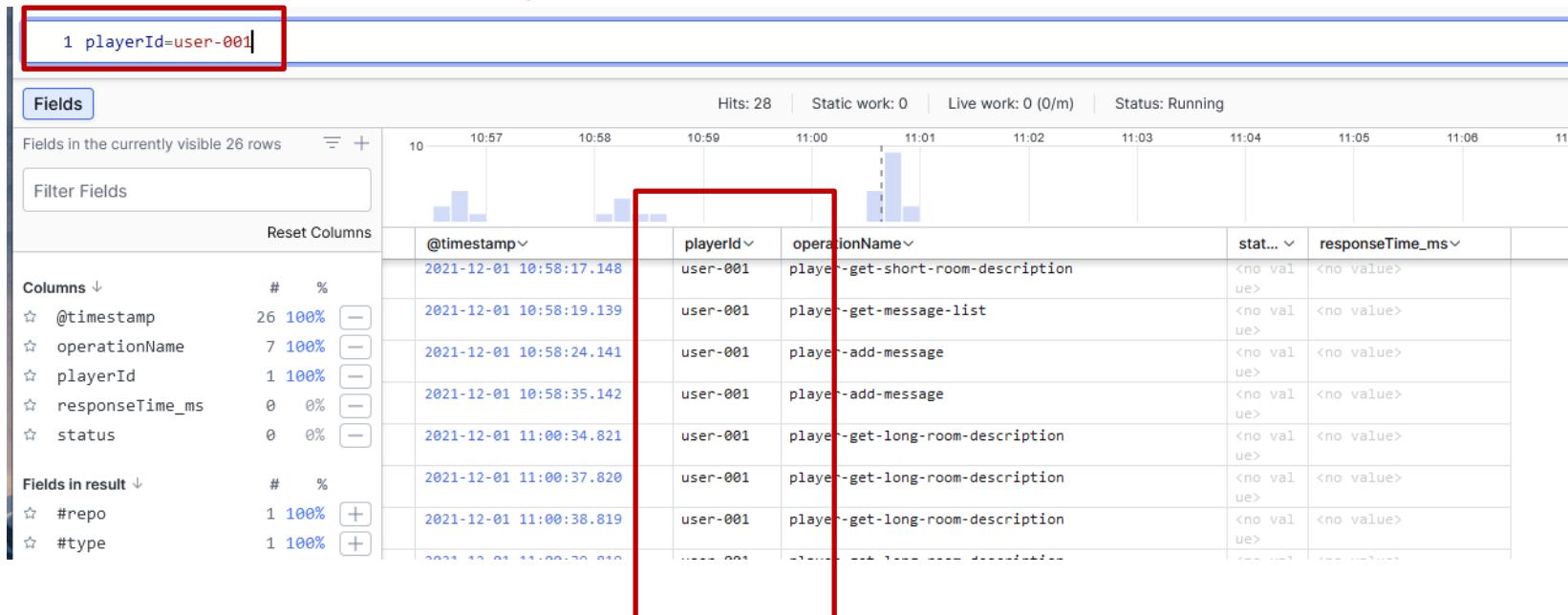


Meta: gradle cmd –Ppcf=malkia.cpf

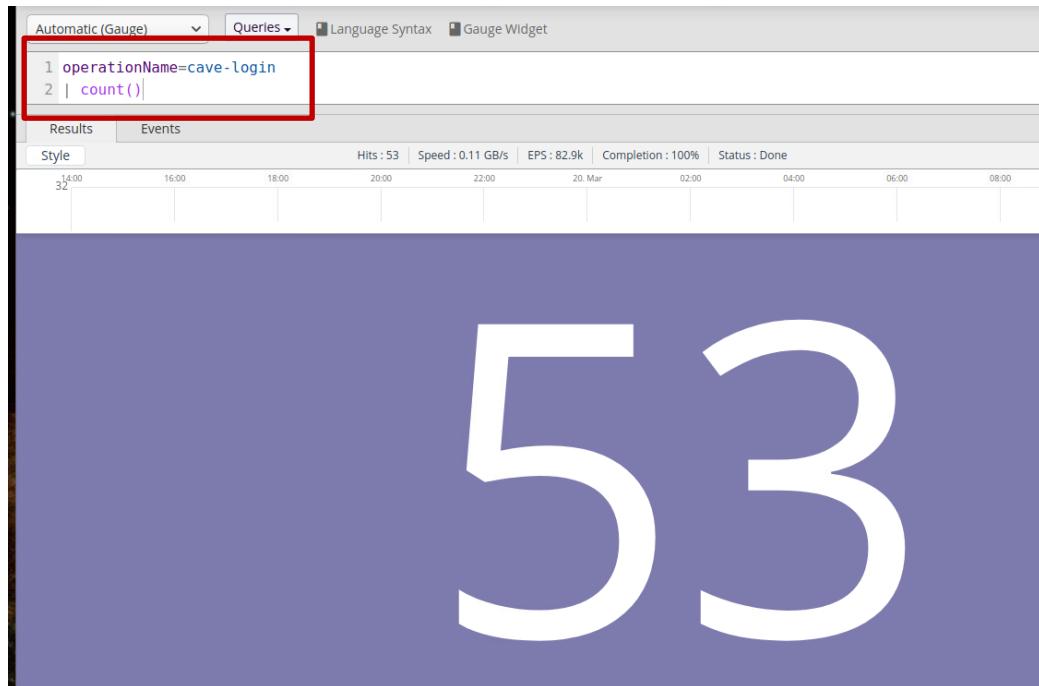
Selecting Columns



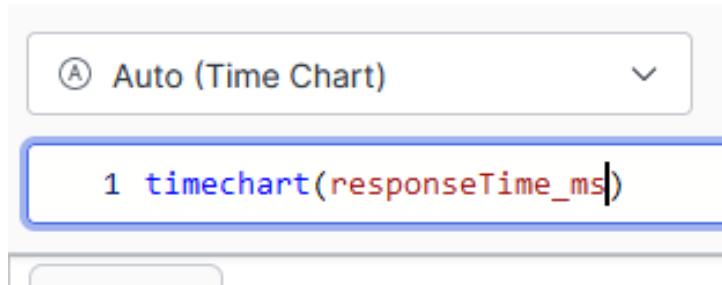
- You can use a query language
 - Do the tutorial ☺



Semi SQL like 😊

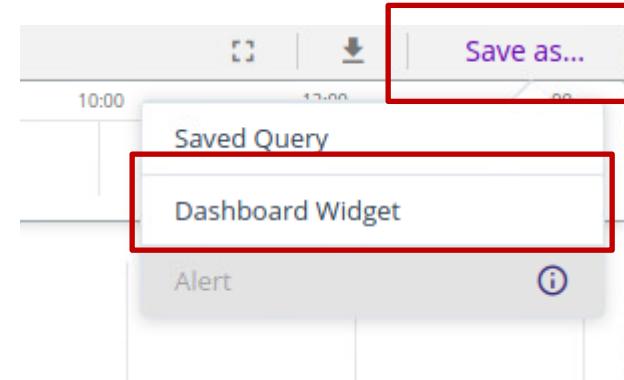


- Perform a search



The screenshot shows a search interface with a dropdown menu set to 'Auto (Time Chart)'. Below the menu is a query input field containing the text '1 timechart(responseTime_ms)'. The interface has a clean, modern design with a light gray background and blue highlights.

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

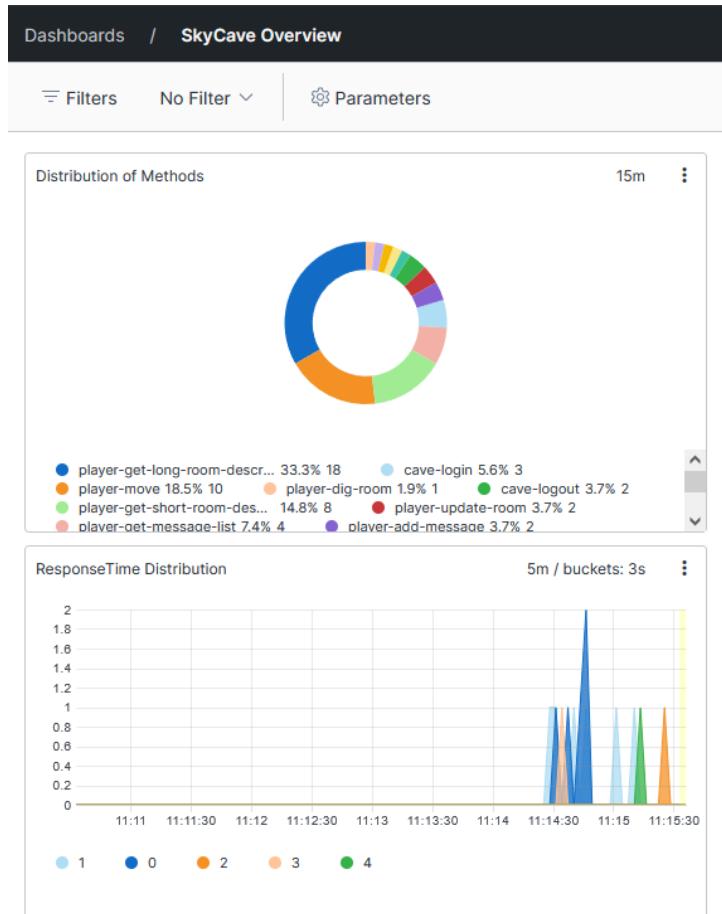


Dashboard Example

- Here I have added two widgets

- Distribution of Methods

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





Swarm

More fiddling ☹



- 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:

```
$ FOO=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://m1-
```

And more ☺

Summary

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