



# Lessons From Using Python GraphQL Libraries to Develop an EPICS PV Server for Web UIs

---

Rebecca Auger-Williams, Observatory Sciences Ltd, St Ives, UK



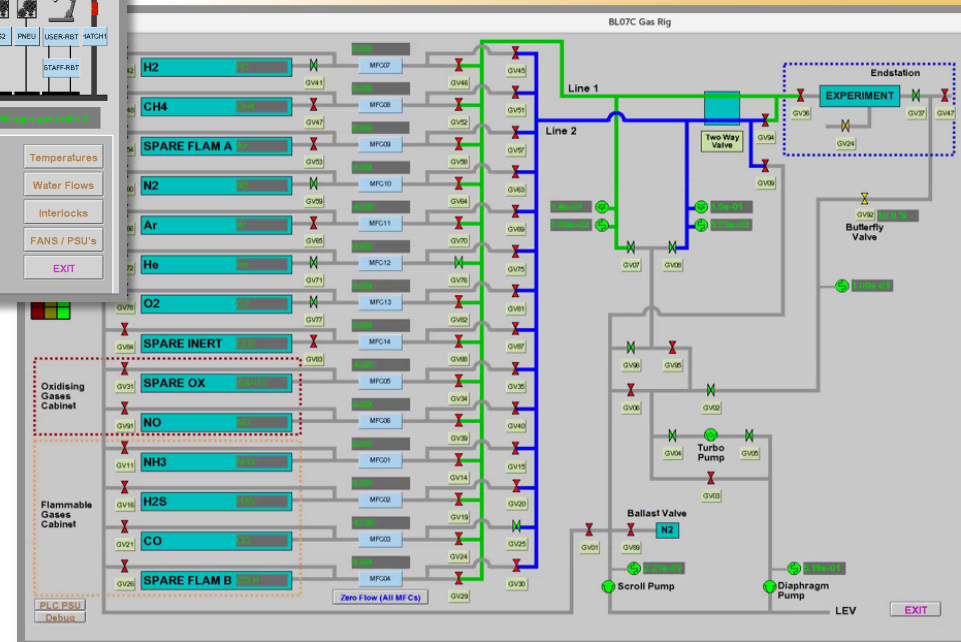
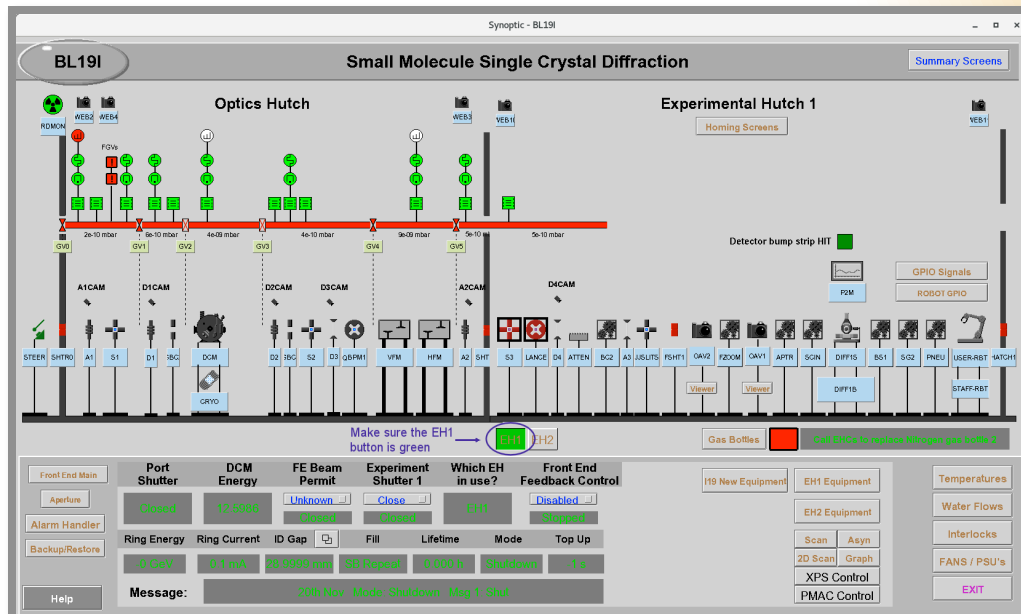
Abigail Alexander, Tom Cobb, Martin Gaughran, Austen Rose, Alexander Wells,  
Andrew Wilson,

Diamond Light Source, Harwell, UK



# Motivation

- Diamond II upgrade prompted UI review





# Motivation

- Existing and potential solutions:

## EDM

- Old technology becoming difficult to deploy

## CS-Studio

(Eclipse based)

- Deprecated
- Heavyweight
- Complex build system



## Phoebus

- Existing solution



## Web UI

- Truly cross-platform
- No installation
- Best experience for remote usage





# Motivation

- Existing and potential solutions:

## EDM

- Old technology becoming difficult to deploy

## CS-Studio

(Eclipse based)

- Deprecated
- Heavyweight
- Complex build system



## Phoebus

- Existing solution



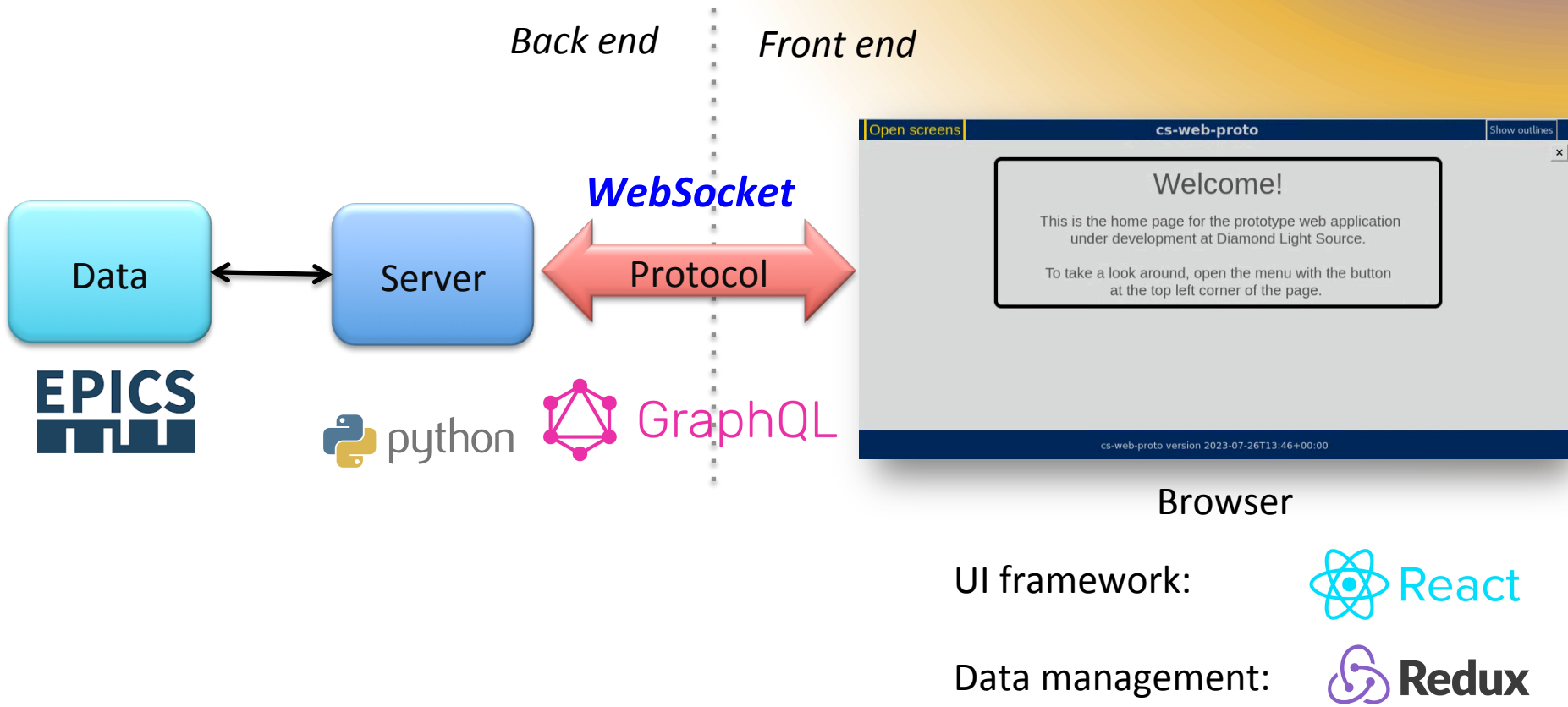
## Web UI

- Truly cross-platform
- No installation
- Best experience for remote usage




# Prototype Web UI

- Introducing **cs-web-proto** created in 2021



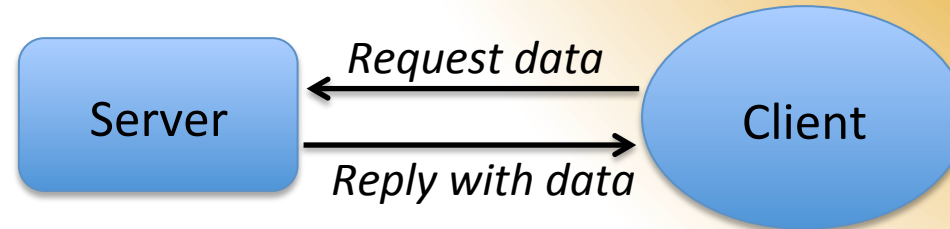
# Back-end Server

- Introducing Coniql 
  - Python application originally developed by Tom Cobb, DLS
  - Uses EPICS Python libraries to access PV data = **aioca**
    - *Built on top of asyncio*
    - API calls: `caget`, `caput`, `camonitor`, `cainfo`
  - GraphQL Python library to serve data to the web UI via WebSockets = **Strawberry GraphQL**



# GraphQL

- Open source query language & runtime engine, originally developed by Facebook
- Client-server model



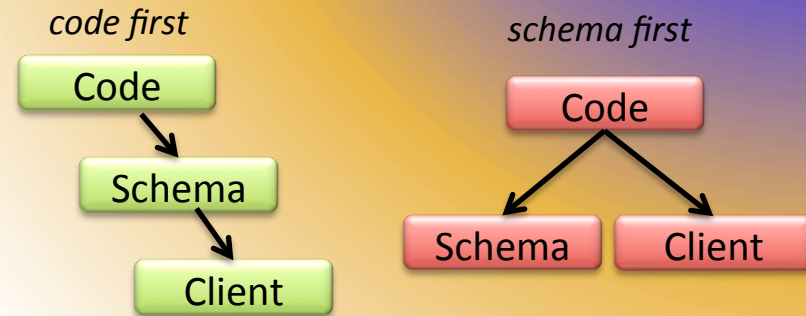
- Supports:
  - Query -> get read-only data
  - Mutation -> modify data
  - Subscription -> receive event-based updates
- Performance and flexibility focused



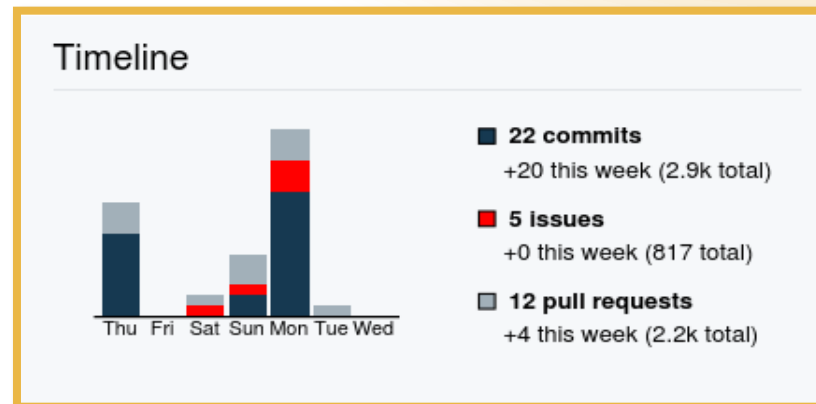
# Strawberry GraphQL



- ✓ Supports code-first schema:
  - Types
  - Resolver functions



- ✓ Supports both new (`graphql-transport-ws`) and deprecated (`graphql-ws`) WebSocket protocols
- ✓ Open source – GitHub
- ✓ In development phase, actively maintained





# Simple Subscription Example

Client makes this request through a WebSocket

Data returned for each field

```
1 subscription {
2   subscribeChannel(id: "ca://temperature:water") {
3     id
4     time {
5       datetime
6     }
7     value {
8       string
9       float
10    }
11    display {
12      units
13      controlRange {
14        max
15        min
16      }
17    }
18  }
19 }
20
```



```
▼ {
▼   "data": {
▼     "subscribeChannel": {
▼       "id": "ca://temperature:water",
▼       "time": {
▼         "datetime": "2023-09-20T14:21:17.939171"
▼       },
▼       "value": {
▼         "string": "50.50",
▼         "float": 50.5
▼       },
▼       "display": {
▼         "units": "C",
▼         "controlRange": {
▼           "max": 100,
▼           "min": 0
▼         }
▼       }
▼     }
▼   }
▼ }
```



# Issues Using GraphQL Libraries

## Issues

Original GraphQL library used was **not well maintained**:

- Memory leak
- No support for new WebSocket protocol

## Solutions

**Refactored** Coniql to use Strawberry



# Issues Using GraphQL Libraries

Issues	Solutions
<p>Original GraphQL library used was <b>not well maintained</b>:</p> <ul style="list-style-type: none"><li>➤ Memory leak</li><li>➤ No support for new WebSocket protocol</li></ul>	<p><b>Refactored</b> Coniql to use Strawberry</p>
<ul style="list-style-type: none"><li>➤ <b>Memory leak</b> in Strawberry for the new WebSocket protocol</li><li>➤ <b>Performance issues</b> (CPU usage) with the new WebSocket protocol</li></ul>	<p>Proposed <b>solution</b>, discussed, fixed and new release – all within days</p>



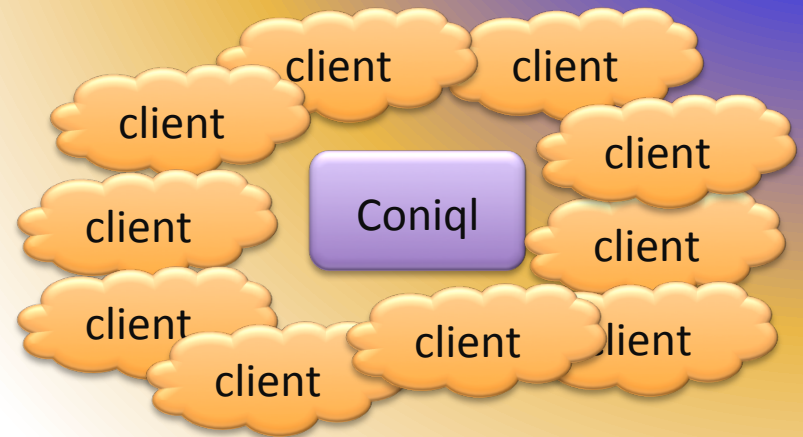
# Issues Using GraphQL Libraries

Issues	Solutions
<p>Original GraphQL library used was <b>not well maintained</b>:</p> <ul style="list-style-type: none"><li>➤ Memory leak</li><li>➤ No support for new WebSocket protocol</li></ul>	<p><b>Refactored</b> Coniql to use Strawberry</p>
<ul style="list-style-type: none"><li>➤ <b>Memory leak</b> in Strawberry for the new WebSocket protocol</li><li>➤ <b>Performance issues</b> (CPU usage) with the new WebSocket protocol</li></ul>	<p>Proposed <b>solution</b>, discussed, fixed and new release – all within days</p>
<p><b>Compatibility</b> issues with new releases</p>	<p>Pin Strawberry <b>version</b> in installation</p>

# Performance Test Objectives

- Demands on Coniql are high!

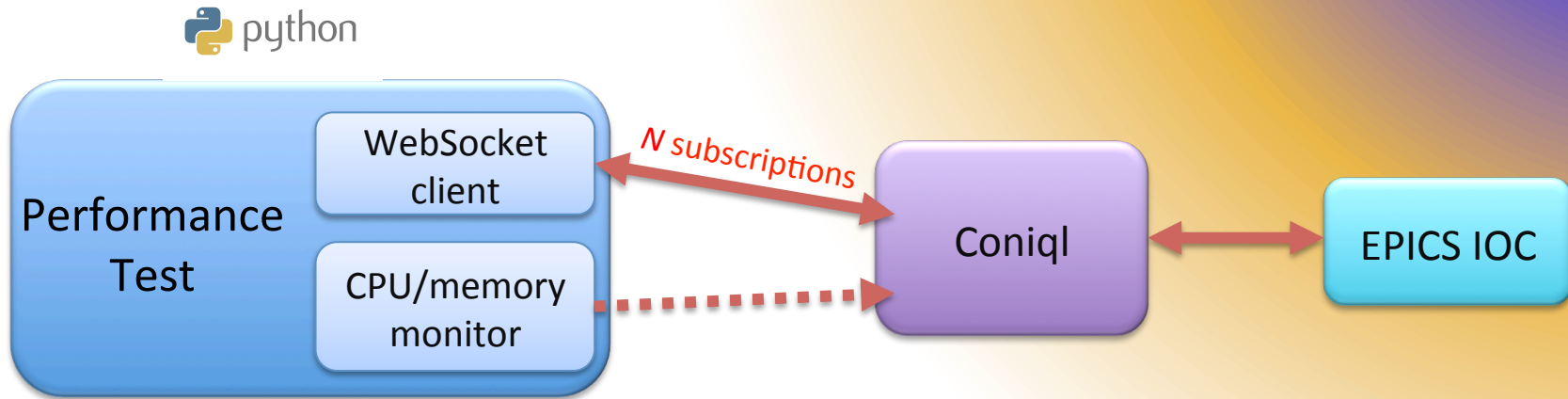
- 1000s of PVs
- Rate up to 10 Hz
- 100s of clients



- Needs to support:

- machine status displays
- operator screens across multiple beamlines
- remote access to these screens

# Performance Test Procedure



## Configurations:

- Number of PVs to subscribe to
- Update frequency (10 Hz)
- Number of incremental updates to collect
- Number of clients

## Measurements:

- CPU usage
- Memory usage
- Number of dropped updates per subscription



# Performance Test Results

- Run tests on Kubernetes (reproducible)
- Test parameters: PVs updating at **10 Hz**, collecting **36,000** samples



# Performance Test Results

- Run tests on Kubernetes (reproducible)
- Test parameters: PVs updating at **10 Hz**, collecting **36,000** samples

Number of clients	Number of PVs	Average CPU	Average number of dropped results
1	10	20.52%	0
1	50	53.32%	0
1	100	74.55%	0
1	200	100.00%	3,811
1	500	100.00%	101,829
2	50	73.70%	0
10	10	75.47%	0

- Slightly disappointing...





# Performance Test Results

- Run tests on Kubernetes (reproducible)
- Test parameters: PVs updating at **10 Hz**, collecting **36,000** samples

Number of clients	Number of PVs	Average CPU	Average number of dropped results
1	10	20.52%	0
1	50	53.32%	0
1	100	74.55%	0
1	200	100.00%	3,811
1	500	100.00%	101,829
2	50	73.70%	0
10	10	75.47%	0

- Slightly disappointing...



100 PVs = OK



# Performance Test Results

- Run tests on Kubernetes (reproducible)
- Test parameters: PVs updating at **10 Hz**, collecting **36,000** samples

Number of clients	Number of PVs	Average CPU	Average number of dropped results
1	10	20.52%	0
1	50	53.32%	0
1	100	74.55%	0
1	200	100.00%	3,811
1	500	100.00%	101,829
2	50	73.70%	0
10	10	75.47%	0

- Slightly disappointing...



100 PVs = OK



At 100% CPU, updates are dropped



# Performance Improvements

- Used sampling profiler to identify problems  
(mostly unnecessary `async` processing & duplicate data for multiple clients)



# Performance Improvements



- Used sampling profiler to identify problems  
(mostly unnecessary async processing & duplicate data for multiple clients)

Number of clients	Number of PVs	Average CPU	Average number of dropped results	Average CPU	Average number of dropped results
1	10	20.52%	0	13.69%	0
1	50	53.32%	0	34.53%	0
1	100	74.55%	0	57.91%	0
1	200	100.00%	3,811	92.60%	16
1	500	100.00%	101,829	100.00%	66,929
2	50	73.70%	0	51.88%	0
10	10	75.47%	0	51.73%	0

**After performance improvements**



# Performance Improvements

**LASTEST RESULTS**

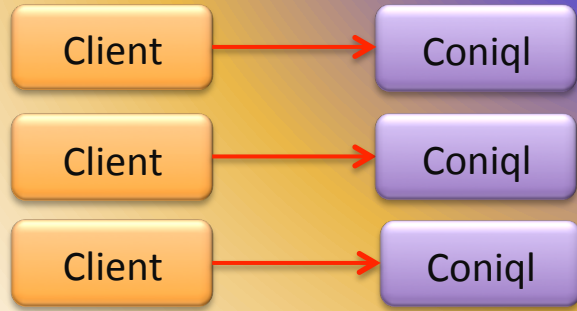
- Used sampling profiler to identify problems (mostly unnecessary async processing & duplicate data for multiple clients)

Number of clients	Number of PVs	Average CPU	Average number of dropped results	Average CPU	Average number of dropped results
1	10	20.52%	0	13.69%	0
1	50	53.32%	0	34.53%	0
1	100	74.55%	0	57.91%	0
1	200	100.00%	3,811	92.60%	16
1	500	100.00%	101,829	100.00%	66,929
2	50	73.70%	0	51.88%	0
10	10	75.47%	0	51.73%	0

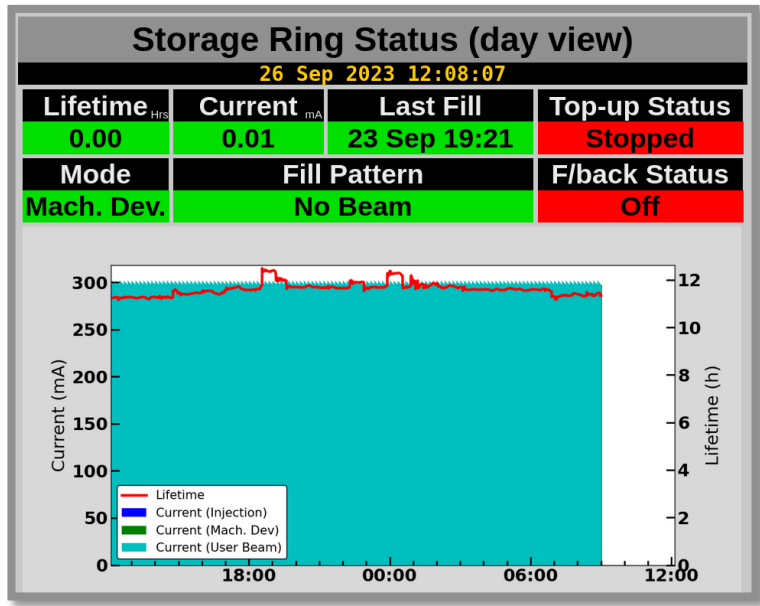
**After performance improvements**

# Kubernetes Deployment

- Mitigate the performance issue using Kubernetes:
  - Deploy 8 replicas -> client connections are load balanced



- Currently this can comfortably support ~ 50 machine status displays



### ID & Front End Status

20 Sep 2023 14:44:33

Lifetime	11.01 h		Current	300.01 mA		
B/L	Gap	Field	Port	Optics	Exptl	
I02	5.1	0.86	Open	Open	Open	
J02	5.0	0.87	Open	Open	Open	
I03	6.8	0.63	Open	Open	Open	
I04	5.0	0.87	Open	Open	Closed	
J04	8.6	0.00	Open	Open Open	Open	
I05	50.0	0.44	Open	Open Closed	Open	
I06	37.6 100.0	0.34 0.04	Open	Open	Closed	Open
I07	6.0	0.80	Open	Open	Closed	
B07			Open	Open		
I08	21.6	0.61	Open	Open	Open	
I09	7.0	0.77	Open	Closed	Open	
J09	50.0	0.13	Open	Closed	Open	
I10	200.0 21.0	0.00 0.56	Open	Open	Open	Closed
I11	5.5	0.79	Open	Open	Open	
K11	22.0	1.54	Open	Open		
I12		4.20	Open	Closed	Closed	
I13	5.9	0.74	Open	Open Open	Open	
J13	8.5	0.55	Open	Open Open	Open	
I14	6.1	0.75	Open	Open	Open	



# Future Plans

- Currently need many Kubernetes replicas...  
... large amount of resources
- Investigate schema changes to increase throughput in low level code (further improve performance?)
- Consider alternatives to GraphQL...  
... PV WebSocket (pvws) ?