A small horizontal bar with a teal segment on the left and an orange segment on the right.

# **IoT Platform for data processing and access.**

Master Practical Course WS 2017

# Contents



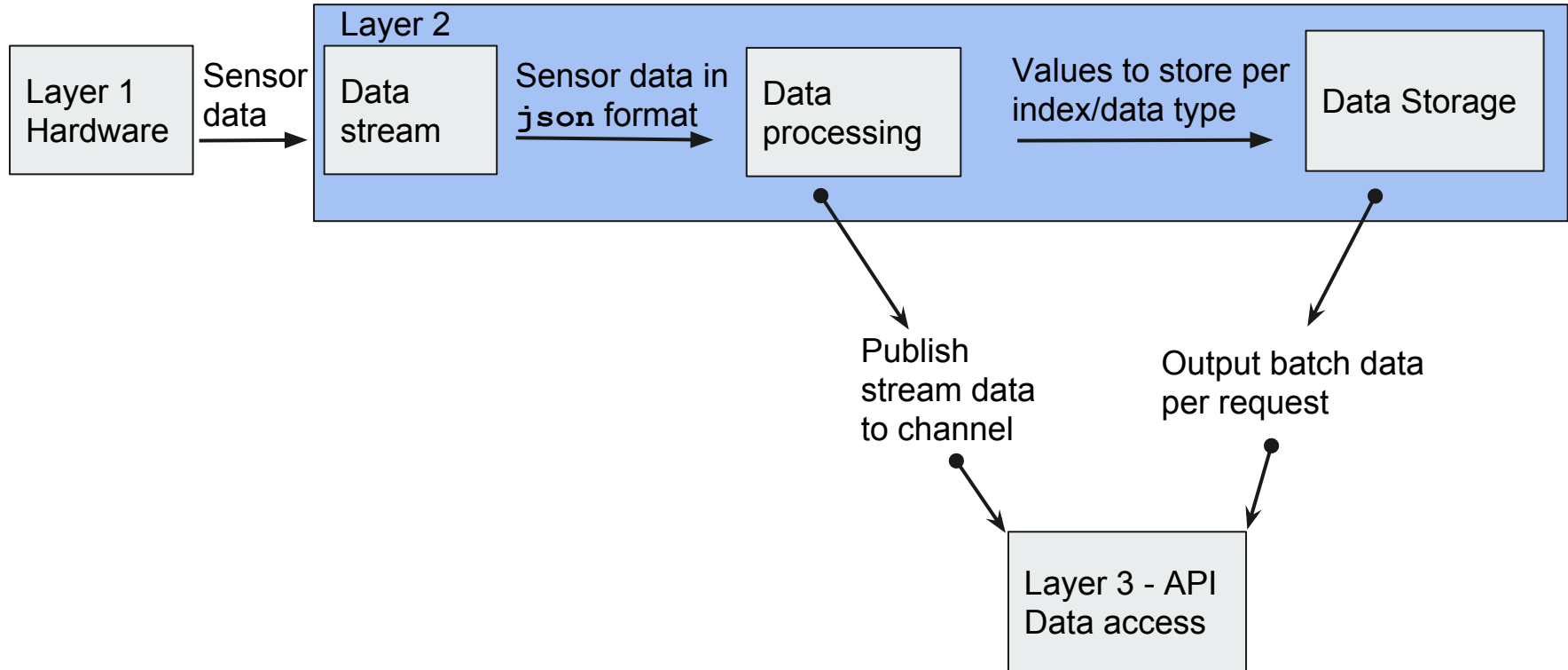
- 0. Introduction
- 1. Sensors
- 2. Data Storage & Processing
- 3. API and Data Access
- 4. Performance Testing

# Goals to Achieve

---

1. **Out-of-the-Box** Highly-Available & Fault-Tolerant system that can collect, store and provision data from sensors in a unified format.
2. Guaranteed **single** message delivery.
3. Possibility to **permanently** store the data.
4. High-speed data **compression**.
5. **Secure** data access using User Authentication.
6. Collect **performance data** and visualize it.

# Information flow through the system



A small horizontal bar with a teal-to-orange gradient is located above the title.

# Sensor Layer

- Erkin Kirdan
- Mikayil Murad
- Hakan Uyumaz
- Ali Naci Uysal

# Overview

## Team I:

Responsible for the development of the on-board part of the platform.

## Goals:

- Support data transmission for as many different boards and sensors out-of-the-box as possible
- Support easy deployment and setup



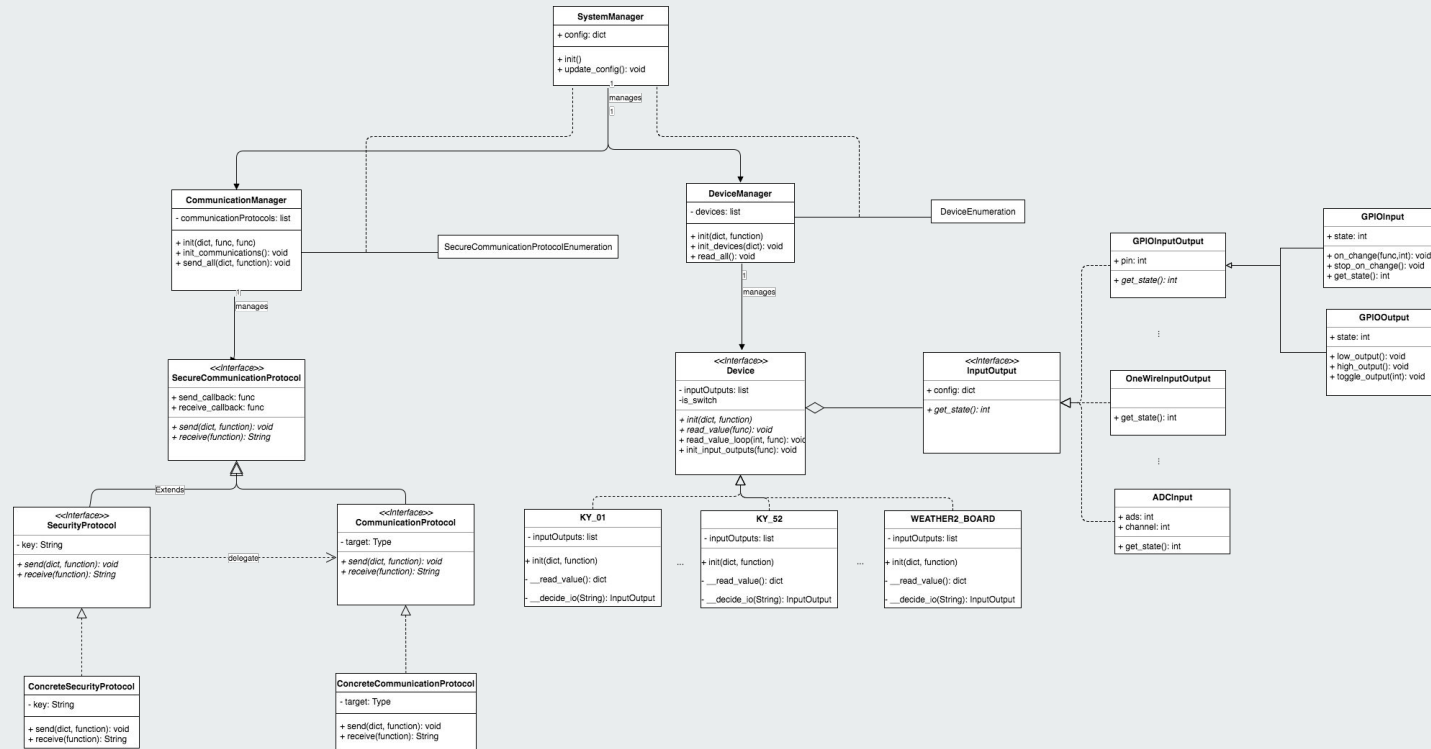
Raspberry PI 3



Odroid-XU 40



# Architecture



# MEAN Stack Web Application

IoT Project Web Server [Home](#) [Register Device](#) [All Devices](#) [Logout](#)

## Register Device

**Unique Device Name**

**Device Description**

**Board Type**

Raspberry Pi

**Created By**

[Register Device](#)

```

▼ 1:
  board_type: "raspberrypi"
  communication_protocols: Array(1)
  ▶ 0: {bootstrap_servers: Array(3), api_version: 10, id: "5a763a93978bbe39c22ceed6", id: "4913c932-8c8a-4d07-9b45-4b6b35bc65eb", device_id: "8b819766-aebb-403b-8fde-9a2e807874f2", length: 1}
    ▶ __proto__: Array(0)
  created_by: "hakan"
  description: "Breadboard with KY02, KY10 and KY26"
  devices: Array(3)
  ▶ 0: {input_output: Array(1), id: "5a763a93978bbe39c22ceed6", id: "0cf0d6c7-6627-4973-98f0-03324d478977", type: "KY02", interval: 5}
  ▶ 1: {input_output: Array(1), id: "5a763a93978bbe39c22ceed4", id: "8a448617-6aee-4c25-94c0-962f563e48ee", type: "KY10", interval: 5}
  ▶ 2: {input_output: Array(2), id: "5a763a93978bbe39c22ceed1", id: "7aa6fb1f-8c9c-4a0d-8c36-20f61ebd9da4", type: "KY26", interval: 5}
    length: 3
    ▶ __proto__: Array(0)
  id: "8b819766-aebb-403b-8fde-9a2e807874f2"
  log_directory: "/var/log/iot/"
  log_level: 0
  name: "Raspberrypi #2"
  __v: 6
  
```

**Device Description**

**Board Type**

Raspberry Pi

**Log Level**

None

**Communication Protocols** [Add Protocol](#)

**Protocol # 1**

Security Type: PlainText | Api Version: 10 | Communication Type: Kafka | Kafka Topic: sensor-input

**Bootstrap Servers** [Add Server](#)

Bootstrap Server IP Address: 141.40.254.141 | Bootstrap Server Port: 9092

**Sensors Wiki** [Add Sensor](#)

**Sensor # 1**

Device Type: Infrared Transmitter module | Read Interval (in seconds): 5

**Input/Output Types** [Add IO](#) [Remove All IO](#)

Type	Name	Pin	Slave Name	Base Directory
One Wire Input/Output	one_wire_io	6	w1_slave	/sys/bus/w1/device
GPIO Input	gpio_input	8		



# Results



1. Provision of the data messages from sensors in the unified format including timestamps and values of the corresponding metrics
2. Basic out-of-the-box-deployed drivers for different types of single-board computers with different types of sensors connected
3. Adjustments of the layer configuration via settings files
4. Additionally built web application for easy configuration
5. Basic data pre-processing according to settings (e.g. use every n-th measure)
6. Easy to extend architecture for new types of boards and sensors

A small decorative horizontal bar with a teal segment on the left and an orange segment on the right.

# Data storage & processing layer

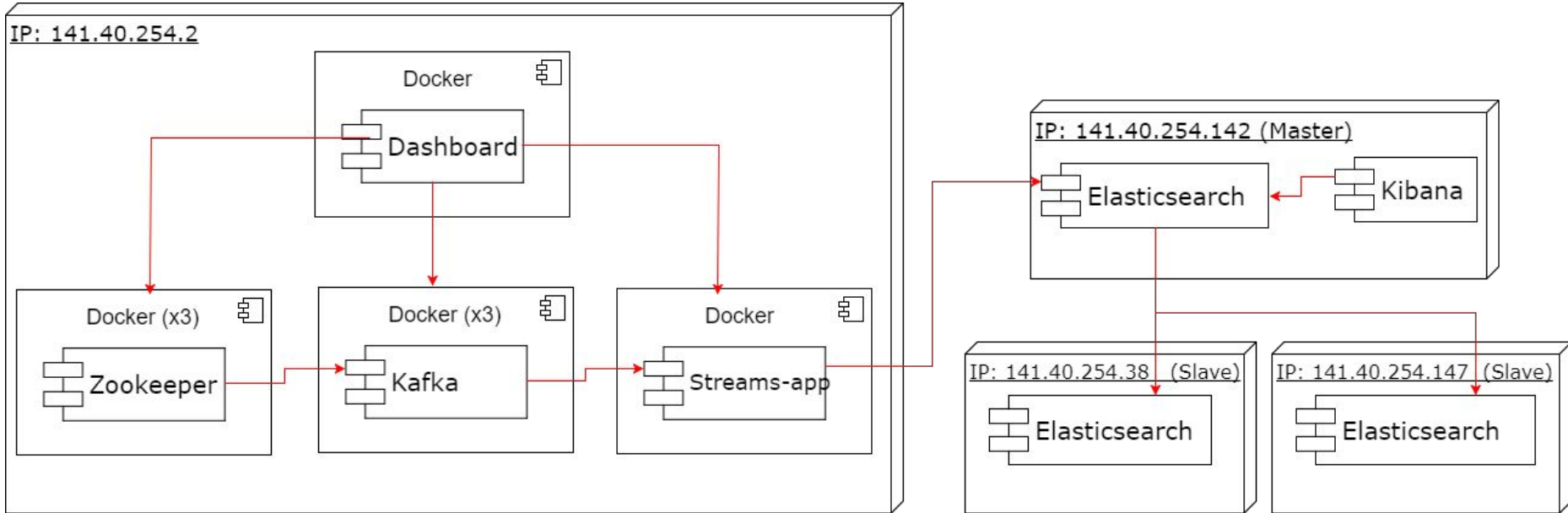
- Yesika Ramirez
- Moawiah Assali
- Atakan Yenel
- Sergey Nasonov

# Contents



- Deployment Diagram
- Configurations
- Streaming & Processing
- ElasticSearch
- Advantages and Challenges

# Deployment Diagram



# Configuration

- Input Channel
- Output Channel
- Database Index
- Database Document type
- The running process
- Kafka broker IP
- Elasticsearch IP

← → ↺ ⌂

141.40.254.2

☆ 🔄 📄 ⋮

CONTAINER		SCHEDULER		TAB-3	
Name	Id	KAFKA IP	ELASTIC IP	KAFKA INPUT CHANNEL	KAFKA OUTPUT CHANNEL
Kafka-Streams-App	caea52ffbc	141.40.254.2	141.40.254.142	sensor-input	streams-sensor-output
kafka-1	a6c1cfb7ce				
kafka-2	192827ac82				
kafka-3	3668ebd5a4				
dashboard	10e1418739				
zookeeper-3	4bb774b987				
zookeeper-2	0749f97c0d				
zookeeper-1	1ab148cd4f				

kafka-input-channel

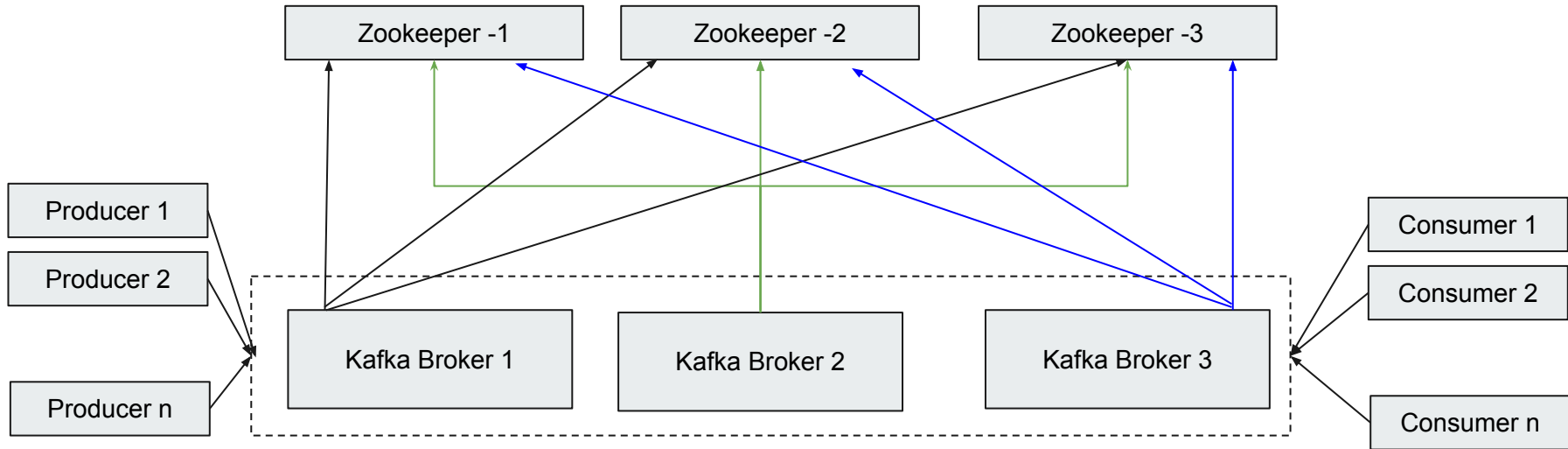
kafka-output-channel

ADD CONTAINER

Container Id

DELETE CONTAINER

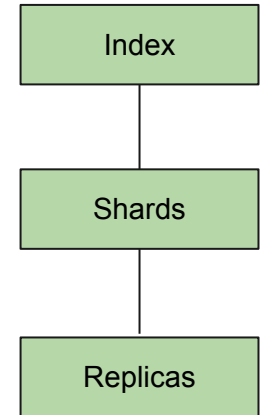
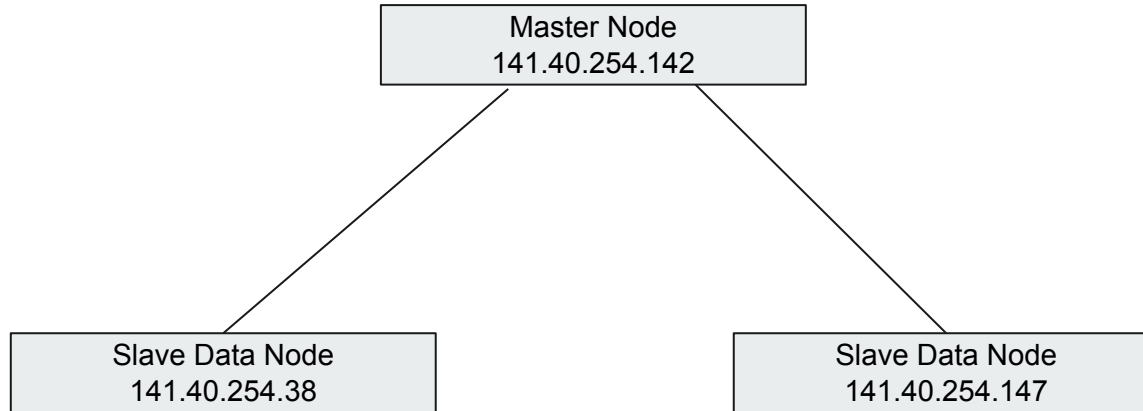
# Streaming & Processing



\*Producer API   \*Consumer API   \*Stream API   \*Connector API

# ElasticSearch

Ports: 9200/9300



\*Master Eligible Node

\*Zen Discovery

\*Election Process



## Advantages

- Fault Tolerance & High Availability
- Scalability of processing engines
- Easy to deploy
- Low cost
- Flexibility via configuration files

## Challenges

- Complexity:
  - Different vendors/versions of the used Software
  - Connectors between streams and ElasticSearch
- Technology Choice
- Security



A small horizontal bar with a teal-to-orange gradient is located above the title.

# API Layer

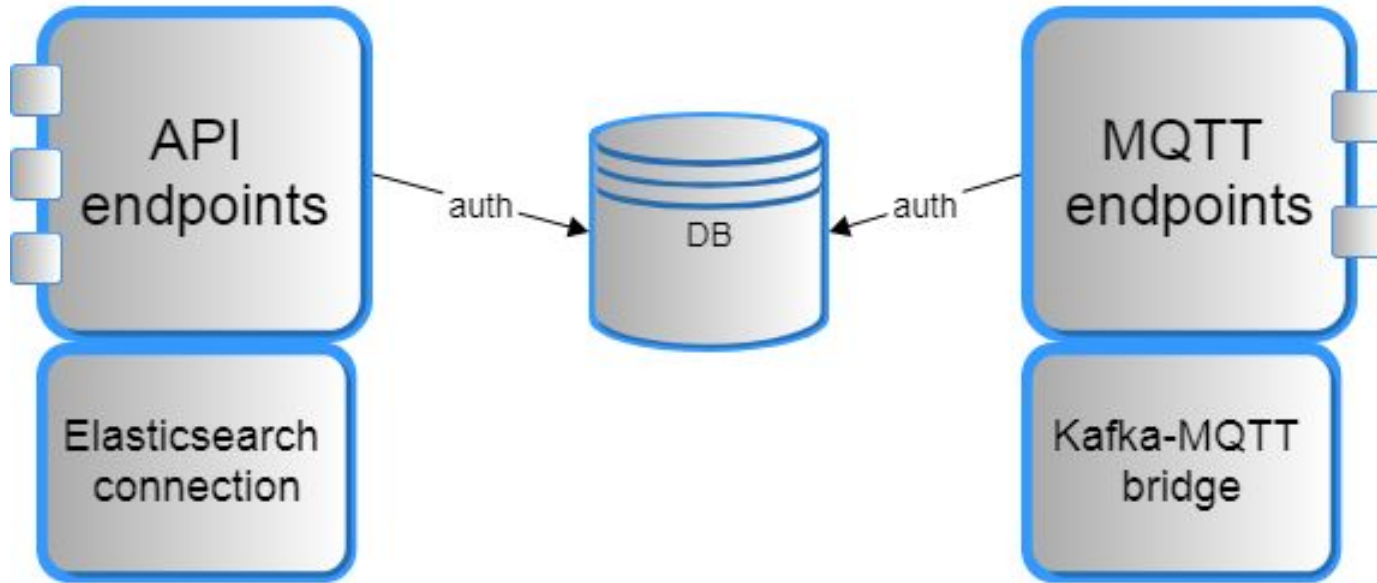
- Sergei Drugalev
- Faisal Hafeez

# Contents

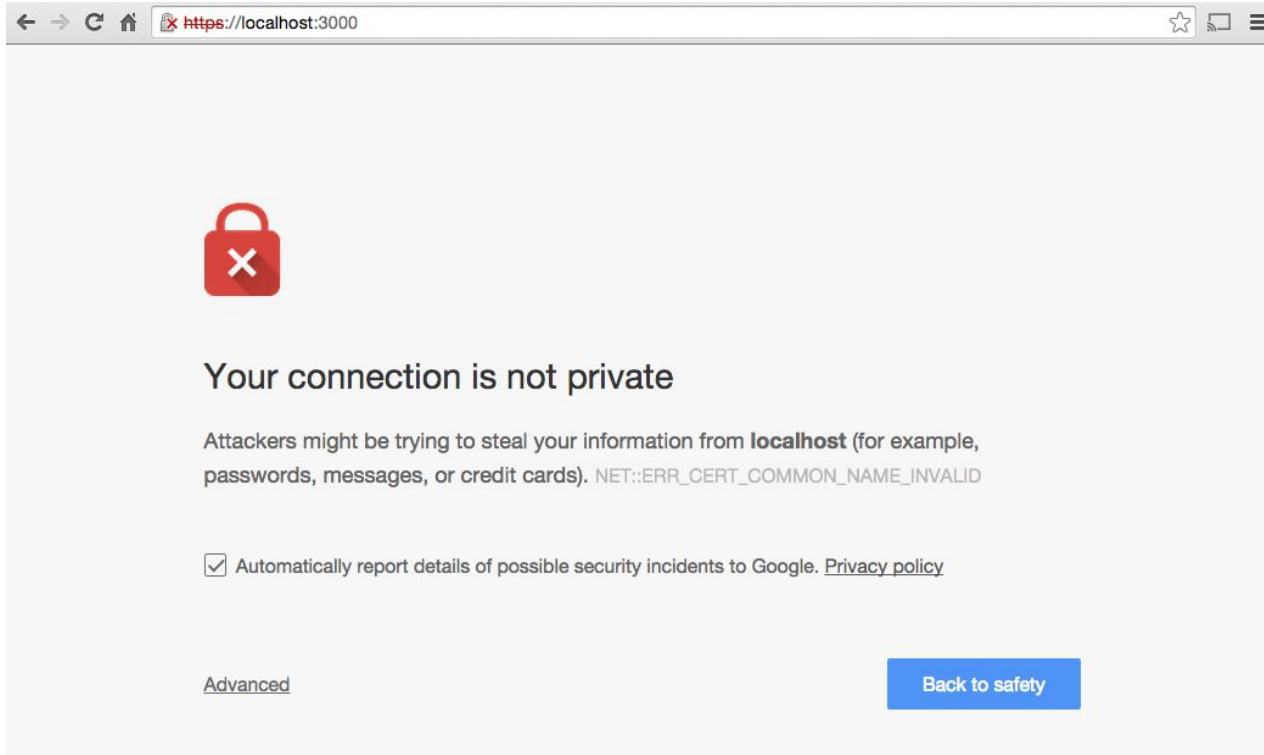


- HTTPS
- User authentication
- User Authorization
- API
- MQTT

# Basic architecture



# HTTPS



# User Authentication

---

- RQlite relational database
- Young project (~1 year old)
- SQLite storage engine
  - Extremely compact and lightweight
  - Doesn't support clustering
  - RQlite does!



# User Authentication



- Basic HTTP authentication (login:password)
- Why not Facebook or Google?
  - Not everybody want/like using their private accounts
  - No web client
  - API server is not a website
  - Still easily doable
- Credentials -> ask your system administrator

# User authorization



- Wait... Is it different?
- Users have configurable access rights
- Ordinary users cannot alter the data

# MQTT

- RABBITMQ broker
- Same authentication AND authorization scheme
- Advantages
  - Lightweight
  - Easy to parse
  - Very little resources on client-side
  - Reliable (QoS 1 and 2)
  - Secure (user authentication + TLS, authorization)
  - Topic wildcards!!



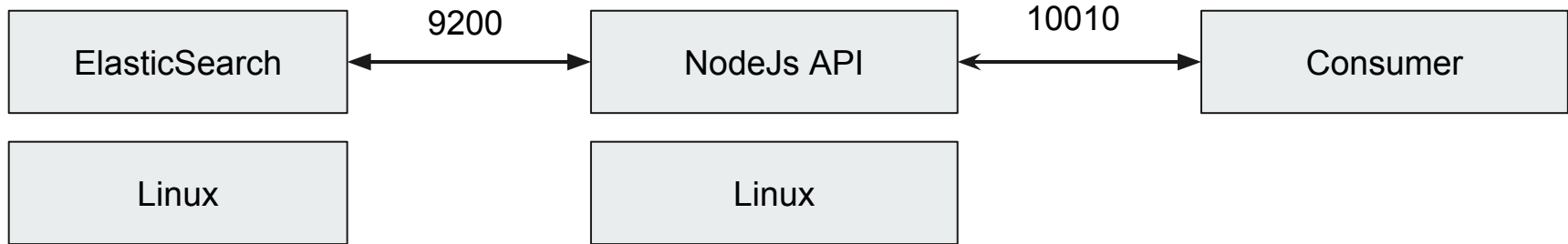


# MQTT - topic structure



- TOPIC
  - /{board ID}/{sensor ID}/{Data Type}
- PAYLOAD
  - {  
    value: {},  
    unit: {}  
}

# API



# API Endpoints

---

- /get\_all\_boards
- /get\_sensor\_data/:board\_id
- /get\_sensor\_data/:board\_id/:sensor\_id
- /get\_sensor\_data/:board\_id/:sensor\_id/:from
- /get\_sensor\_data/:board\_id/:sensor\_id/:from/:to

# Challenges



- Dynamic IPs
- Backup machines
- Elasticsearch server down
- Disk Storage

A small horizontal bar with a teal segment on the left and an orange segment on the right is located above the title.

# Performance Testing

- Shumail Mohyuddin
- Muhammad Razzaque Bhatti
- Zaryab Khan

# Goals



- Monitor the platform and its performance
- Alerts based on threshold values
- Visualize the collected data
- Highlight bottlenecks and recommend possible improvements

# Technology stack

---

- **K6**

Generate load via virtual users - Behavior test cases written in JavaScript

Test platform performance under stress

- **InfluxDB, Prometheus**

Time based database and metrics storage from multiple sources

- **Grafana**

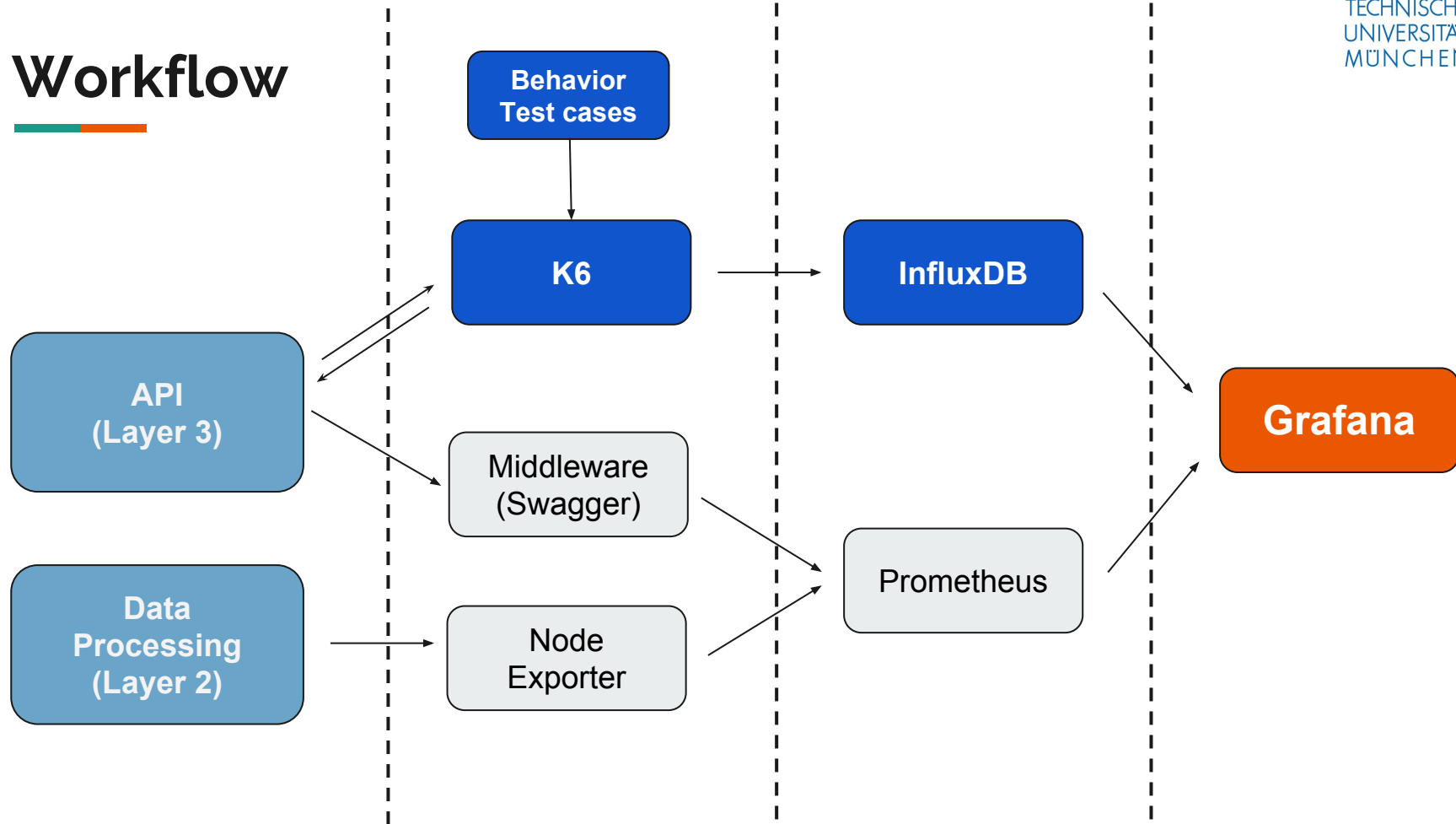
Data visualization across multiple sources

- **Swagger API Framework middlewares with ExpressJS**

Intercepting the API requests for event logging

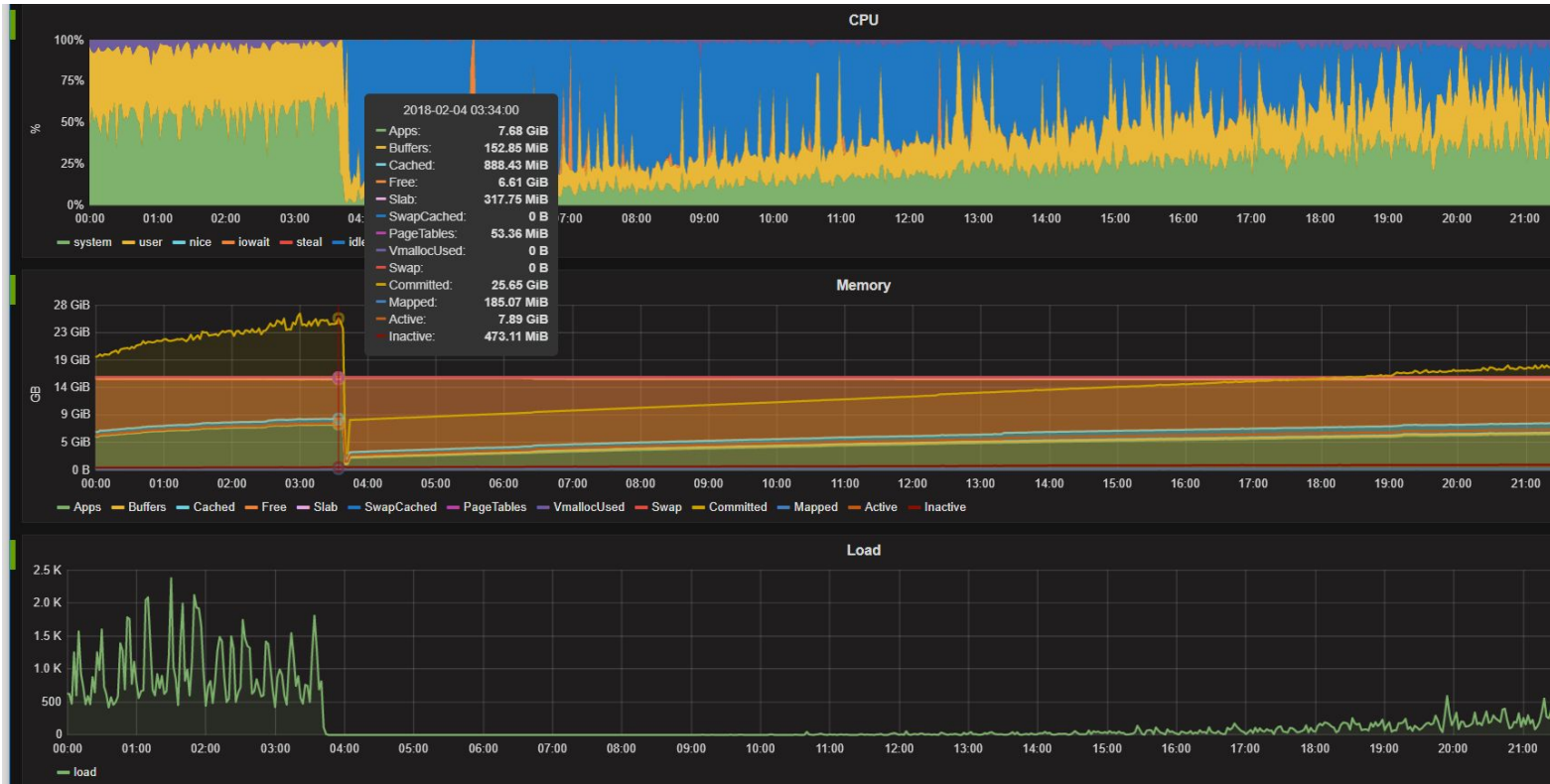
- **Node Exporter** - Infrastructure monitoring and alerts

# Workflow





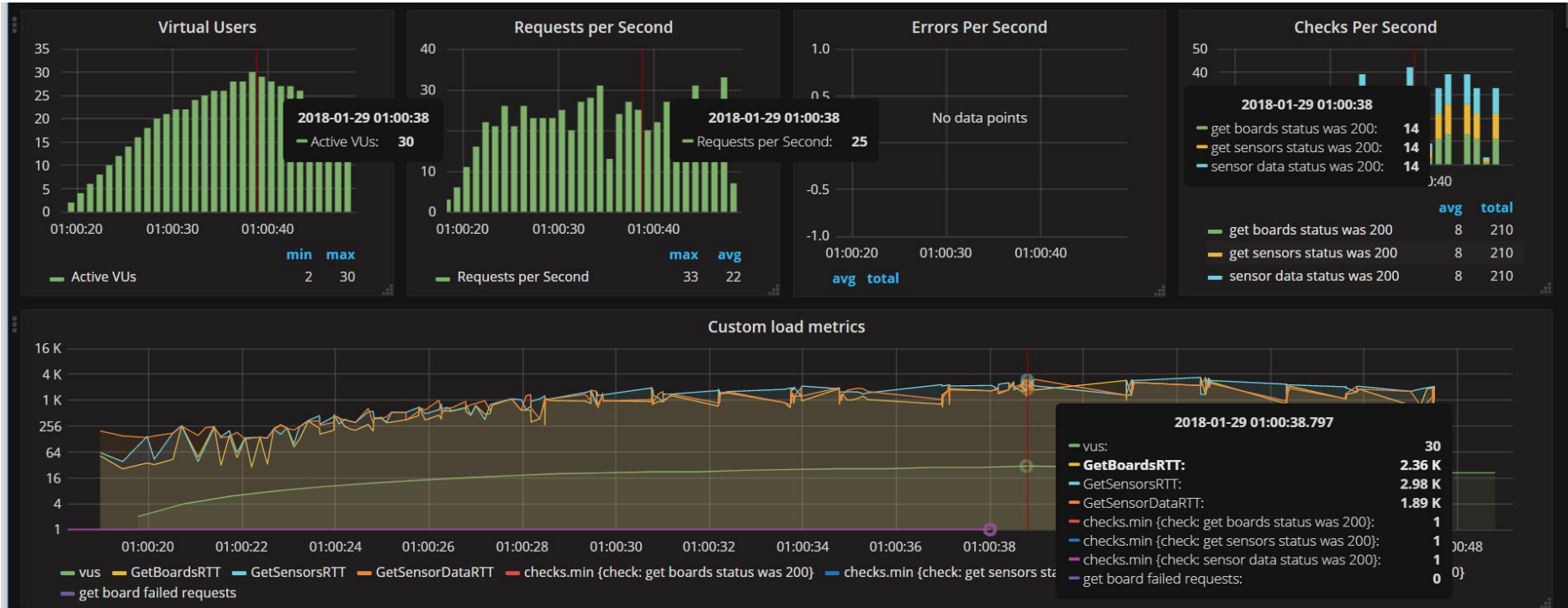
# Data storage and processing monitoring



# Stress Testing

- Virtual Users - 35
  - Increased Dynamically over time
  - Behavior:
    - First request on to get list of boards
    - Select a board and send second request for getting sensors
    - 3rd Request for fetching values from sensors
    - 2 more requests for fetching sensors data
  - One request sent by each VU for this behavior in every second.
  - ~8000 API calls in 30 seconds
- Results:
  - Alert generated when avg RTT over test > 3s
  - Threshold reached at VU= 30
  - RTT averages from 20 milliseconds to ~3 seconds under max stress.

# Stress Testing



# Monitoring Statistics Interfaces



- **REST API**

GET on `<api-url>/swagger-stats/stats`

- **Prometheus Metrics**

Available on `<api-url>/swagger-stats/metrics`

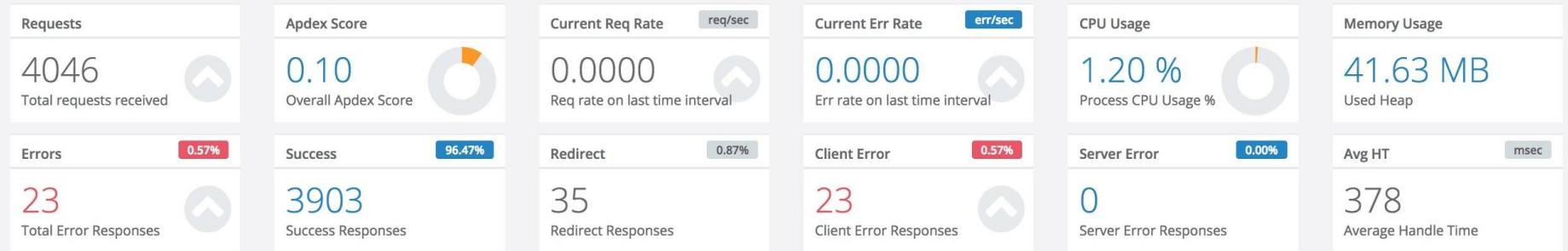
- **Visualisation Dashboard**

Available on `<api-url>/swagger-stats/ui`

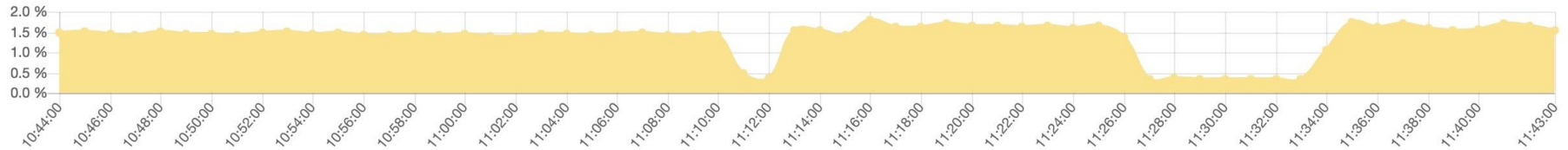
# API Realtime Monitoring



## Summary



CPU Usage % over last 60 minutes



# API Statistics via API

GET
http://141.40.254.131:10010/swagger-stats/stats
Params
Send
Save

Body
Cookies
Headers (6)
Tests
Status: 200 OK Time: 35 ms Size: 958 B

Pretty
Raw
Preview
JSON

```

1- {
2  "startts": 1517179840328,
3  "all": {
4    "requests": 4047,
5    "responses": 3962,
6    "errors": 24,
7    "info": 0,
8    "success": 3903,
9    "redirect": 35,
10   "client_error": 24,
11   "server_error": 0,
12   "total_time": 1531404,
13   "max_time": 30007,
14   "avg_time": 378.4047442550037,
15   "total_req_length": 0,
16   "max_req_length": 0,
17   "avg_req_length": 0,
18   "total_res_length": 74242552,
19   "max_res_length": 276548,
20   "avg_res_length": 18738,
21   "req_rate": 0.006647359855914804,
22   "err_rate": 0.000039420962822326484,
23   "apdex_threshold": 25,
24   "apdex_satisfied": 31,
25   "apdex_tolerated": 740,
26   "apdex_score": 0.10121150933871782
27 },

```

# Performance Recommendations



- Though current platform “survived” and didn’t crash...But.
- More resources (especially compute power) overall
  - CPU usage reached 100% under heavy stress on Layer-2
  - Similar recommendation for Layer-3
- Load-balancer behind Layer-3 (API) Entry point to have more worker nodes for fault-tolerance and scalability.
- Massive Scalability? Microservices architecture

# Live Previews

