Task 1: Sending Formula 1 Car Data to Cloud with Eclipse Kuksa

Basically, this task is about sending data from one application (f1demo2val) to Kuksa Server and then from the server to another application (val2mqtt) which further publish this data as mosquito messages to mosquito broker.

In the below image, I have built first image for f1demo2val application with f1demo2val image name and 1.0 tag:

- ✓ docker build -t docker-registry.rahti.csc.fi/finalprojecttask1/f1demo2val:1.0 -f Dockerfile .

```
C:\Windows\System32\cmd.exe
D:\check\test\module\task1\f1demo2val>docker build -t docker-registry.rahti.csc.fi/finalprojecttask1/f1demo2val:1.0 -f D
ockerfile .
[+] Building 1.6s (2/3)                                                                          docker:default
 => [internal] load build definition from Dockerfile                                                        0.1s
[+] Building 4.0s (10/10) FINISHED                                                               docker:default
 => [internal] load build definition from Dockerfile                                                        0.1s
 => => transferring dockerfile: 588B                                                                        0.0s
 => [internal] load .dockerignore                                                                           0.1s
 => => transferring context: 2B                                                                             0.0se
 => [internal] load metadata for docker.io/library/python:3.9.0-slim                                        3.4s
 => [auth] library/python:pull token for registry-1.docker.io                                               0.0s
 => [1/4] FROM docker.io/library/python:3.9.0-slim@sha256:de8d4a338fb815509de2046cea4ff48959c84aabc1b65cb41c5e2da  0.0s
 => [internal] load build context                                                                           0.1s
 => => transferring context: 3.36kB                                                                         0.1s
 => CACHED [2/4] WORKDIR /app                                                                               0.0s
 => CACHED [3/4] COPY . /app                                                                                0.0s
 => CACHED [4/4] RUN pip install --trusted-host pypi.python.org -r requirements.txt                         0.0s
 => exporting to image                                                                                      0.0s
 => => exporting layers                                                                                     0.0s
 => => writing image sha256:cdee46b2d0c2c90d147927d17fd1736b614fda99cc6bca00fa595276b3ee6e12                0.0s
 => => naming to docker-registry.rahti.csc.fi/finalprojecttask1/f1demo2val:1.0                              0.0s

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview
```

Then I pushed this image to my csc rahti account inside the finalprojecttask1 project:

- ✓ docker push docker-registry.rahti.csc.fi/finalprojecttask1/f1demo2val:1.0

```
C:\Windows\System32\cmd.exe
D:\check\test\module\task1\f1demo2val>docker push docker-registry.rahti.csc.fi/finalprojecttask1/f1demo2val:1.0
[2023-12-18T00:37:02.6374868000Z][docker-credential-desktop.system][W] Windows version might not be up-to-date: The syste
m cannot find the file specified.
The push refers to repository [docker-registry.rahti.csc.fi/finalprojecttask1/f1demo2val]
8c8e14c1a842: Layer already exists
5e97ea1debd5: Layer already exists
1647e1d28bf5: Layer already exists
b25b816984b3: Layer already exists
7f87da26703a: Layer already exists
889e9591fc68: Layer already exists
0d92fd747771: Layer already exists
f5600c6330da: Layer already exists
1.0: digest: sha256:6d53f0d324a9018a106f87a316dacc7a1f60a05fa35743002820f9560a4fceee size: 2001

D:\check\test\module\task1\f1demo2val>
```

Similarly, I built second image for val2mqtt application with val2mqtt image name and 1.0 tag:

- ✓ docker build -t docker-registry.rahti.csc.fi/finalprojecttask1/val2mqtt:1.0 -f Dockerfile .

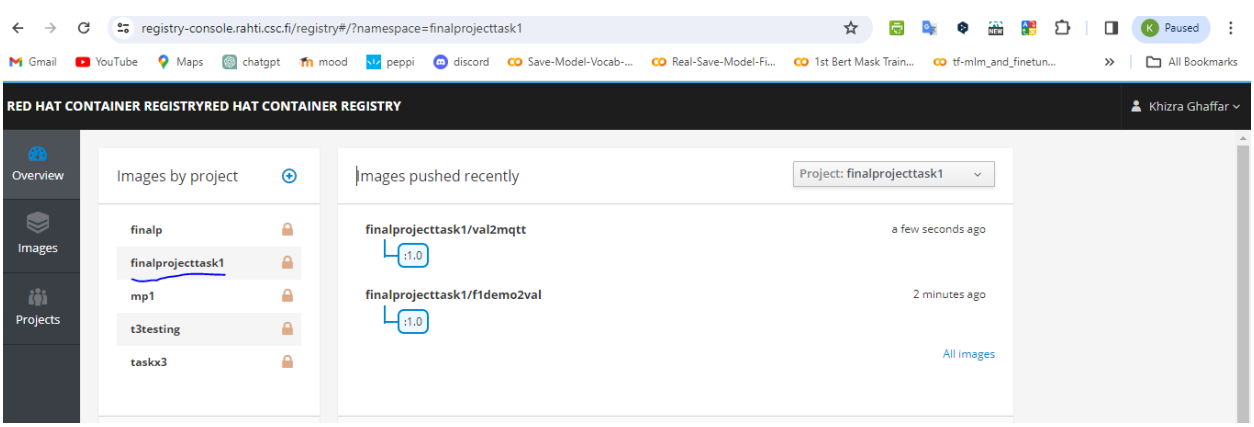
```
C:\Windows\System32\cmd.exe
D:\check\test\module\task1\val2mqtt>docker build -t docker-registry.rahti.csc.fi/finalprojecttask1/val2mqtt:1.0 -f Docke
rfile .
[+] Building 0.2s (2/3)                                                                          docker:default
 => [internal] load build definition from Dockerfile                                                       0.1s
[+] Building 1.2s (9/9) FINISHED                                                                 docker:default
 => [internal] load build definition from Dockerfile                                                       0.1s
 => => transferring dockerfile: 576B                                                                       0.0s
 => [internal] load .dockerignore                                                                          0.1s
 => => transferring context: 2B                                                                            0.0se
 => [internal] load metadata for docker.io/library/python:3.9.0-slim                                       0.8s
 => [1/4] FROM docker.io/library/python:3.9.0-slim@sha256:de8d4a338fb815509de2046cea4ff48959c84aabc1b65cb41c5e2da  0.0s
 => [internal] load build context                                                                          0.1s
 => => transferring context: 2.08kB                                                                        0.0s
 => CACHED [2/4] WORKDIR /app                                                                              0.0s
 => CACHED [3/4] COPY . /app                                                                               0.0s
 => CACHED [4/4] RUN pip install --trusted-host pypi.python.org -r requirements.txt                        0.0s
 => exporting to image                                                                                     0.1s
 => => exporting layers                                                                                    0.0s
 => => writing image sha256:9f33111f3eb048d3eea1be1a510f118ad9d720bed3a14bd1d5e664e9b9a63125               0.0s
 => => naming to docker-registry.rahti.csc.fi/finalprojecttask1/val2mqtt:1.0                               0.0s

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview
```

Again, I pushed this image to my csc rahti account inside the finalprojecttask1 project:

- ✓ docker push docker-registry.rahti.csc.fi/finalprojecttask1/val2mqtt:1.0

```
C:\Windows\System32\cmd.exe
D:\check\test\module\task1\val2mqtt>docker push docker-registry.rahti.csc.fi/finalprojecttask1/val2mqtt:1.0
[2023-12-18T00:39:18.989202100Z][docker-credential-desktop.system][W] Windows version might not be up-to-date: The syste
m cannot find the file specified.
The push refers to repository [docker-registry.rahti.csc.fi/finalprojecttask1/val2mqtt]
afcbdc43485c: Layer already exists
5abb1f05e2ac: Layer already exists
1647e1d28bf5: Layer already exists
b25b816984b3: Layer already exists
7f87da26703a: Layer already exists
889e9591fc68: Layer already exists
0d92fd747771: Layer already exists
f5600c6330da: Layer already exists
1.0: digest: sha256:e5619d297f58bede2e420bf2b94a55a6941bf5215ef16700cc19537adac85259 size: 1998

D:\check\test\module\task1\val2mqtt>
```

CSC Rahit Dashboard:



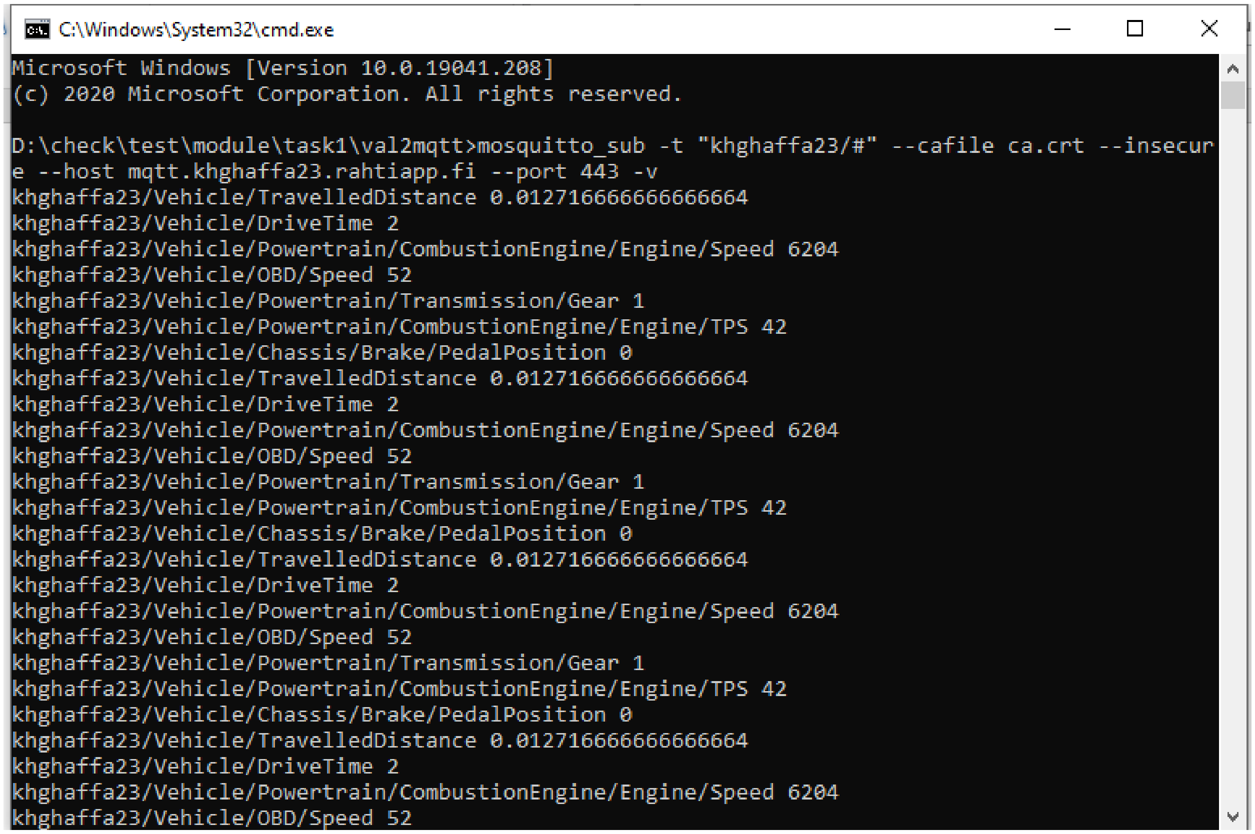
Further, I checked the messages received to the Kuksa Server:

- ✓ Docker-compose up

```
C:\Windows\System32\cmd.exe
D:\check\test\module\task1>docker-compose up
time="2023-12-18T02:42:48+02:00" level=warning msg="a network with name khghaffa23-net exists but was not created for pr
oject \"task1\".\nSet `external: true` to use an existing network"
[+] Running 3/3
 ⠿ Container task1-kuksa.val-1   Created                                                                    0.1s
 ⠿ Container task1-val2mqtt-1    Created                                                                    0.2s
 ⠿ Container task1-f1demo2val-1  Created                                                                    0.2s
Attaching to task1-f1demo2val-1, task1-kuksa.val-1, task1-val2mqtt-1
task1-kuksa.val-1    | Starting kuksa.val
task1-kuksa.val-1    | No config file, initialize with example config
task1-kuksa.val-1    | No VSS tree, initialize with example
task1-kuksa.val-1    | No server keys configured, initialize with example
task1-kuksa.val-1    | No jwt key configured, initialize with example
task1-kuksa.val-1    | kuksa.val server
task1-kuksa.val-1    | Commit 3127002-dirty from 2021-12-16T22:56:19+01:00
task1-kuksa.val-1    | Read configs from /config/config.ini
task1-kuksa.val-1    | Update vss path to /config/vss.json
task1-kuksa.val-1    | Update cert-path to /config/certs
task1-kuksa.val-1    | Log START
task1-kuksa.val-1    | VERBOSE: Try reading JWT pub key from /config/certs/jwt.key.pub
task1-kuksa.val-1    | VERBOSE: SubscribeThread: Started Subscription Thread!
task1-kuksa.val-1    | VERBOSE: VssDatabase::VssDatabase : VSS tree initialized using JSON file = /config/vss.json
task1-kuksa.val-1    | INFO: Setting default for Vehicle/Body/ChargingPort/Type to unknown
task1-kuksa.val-1    | INFO: Setting default for Vehicle/Cabin/DoorCount to 4
task1-kuksa.val-1    | INFO: Setting default for Vehicle/Cabin/DriverPosition to 1
task1-kuksa.val-1    | INFO: Setting default for Vehicle/Cabin/SeatPosCount to [2,3]
task1-kuksa.val-1    | INFO: Setting default for Vehicle/Cabin/SeatRowCount to 2
task1-kuksa.val-1    | INFO: Setting default for Vehicle/Chassis/AxleCount to 2
task1-kuksa.val-1    | INFO: Setting default for Vehicle/Chassis/CurbWeight to 0
task1-kuksa.val-1    | INFO: Setting default for Vehicle/Chassis/GrossWeight to 0
```

In the last, I checked that messages are published to the mosquito broker with mosquito sub command:

- ✓ mosquitto_sub -t "khghaffa23/#" --cafile ca.crt --insecure --host mqtt.khghaffa23.rahtiapp.fi --port 443 -v



```
Microsoft Windows [Version 10.0.19041.208]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\check\test\module\task1\val2mqtt>mosquitto_sub -t "khghaffa23/#" --cafile ca.crt --insecur
e --host mqtt.khghaffa23.rahtiapp.fi --port 443 -v
khghaffa23/Vehicle/TravelledDistance 0.012716666666666664
khghaffa23/Vehicle/DriveTime 2
khghaffa23/Vehicle/Powertrain/CombustionEngine/Engine/Speed 6204
khghaffa23/Vehicle/OBD/Speed 52
khghaffa23/Vehicle/Powertrain/Transmission/Gear 1
khghaffa23/Vehicle/Powertrain/CombustionEngine/Engine/TPS 42
khghaffa23/Vehicle/Chassis/Brake/PedalPosition 0
khghaffa23/Vehicle/TravelledDistance 0.012716666666666664
khghaffa23/Vehicle/DriveTime 2
khghaffa23/Vehicle/Powertrain/CombustionEngine/Engine/Speed 6204
khghaffa23/Vehicle/OBD/Speed 52
khghaffa23/Vehicle/Powertrain/Transmission/Gear 1
khghaffa23/Vehicle/Powertrain/CombustionEngine/Engine/TPS 42
khghaffa23/Vehicle/Chassis/Brake/PedalPosition 0
khghaffa23/Vehicle/TravelledDistance 0.012716666666666664
khghaffa23/Vehicle/DriveTime 2
khghaffa23/Vehicle/Powertrain/CombustionEngine/Engine/Speed 6204
khghaffa23/Vehicle/OBD/Speed 52
khghaffa23/Vehicle/Powertrain/Transmission/Gear 1
khghaffa23/Vehicle/Powertrain/CombustionEngine/Engine/TPS 42
khghaffa23/Vehicle/Chassis/Brake/PedalPosition 0
khghaffa23/Vehicle/TravelledDistance 0.012716666666666664
khghaffa23/Vehicle/DriveTime 2
khghaffa23/Vehicle/Powertrain/CombustionEngine/Engine/Speed 6204
khghaffa23/Vehicle/OBD/Speed 52
```

Task 3: Monitoring Cluster with Prometheus and Grafana

Task 3.1 (A)

I selected the task 3 for project work, this task is about monitoring the pods with Prometheus and Grafana open source tools for analyzing their performance. Basically, Prometheus is acts as a data source and provide data in time series format. On the other hand, Grafana provides the graphical user interface for to user for visualizing the data in the form of graphs, charts and other forms.

Moreover, Grafana uses the Prometheus as a data source and provide visualization of this data in different ways. It can use different data sources such as databases, cloud databases, InfluxDB and many others.

I deployed the Prometheus and Grafana using the rahti template:

- ✓ oc new-app -f ./prometheus-grafana.yaml -p NAMESPACE=t3testing

In the below picture, there is a success message which shows that resources have been created successfully:

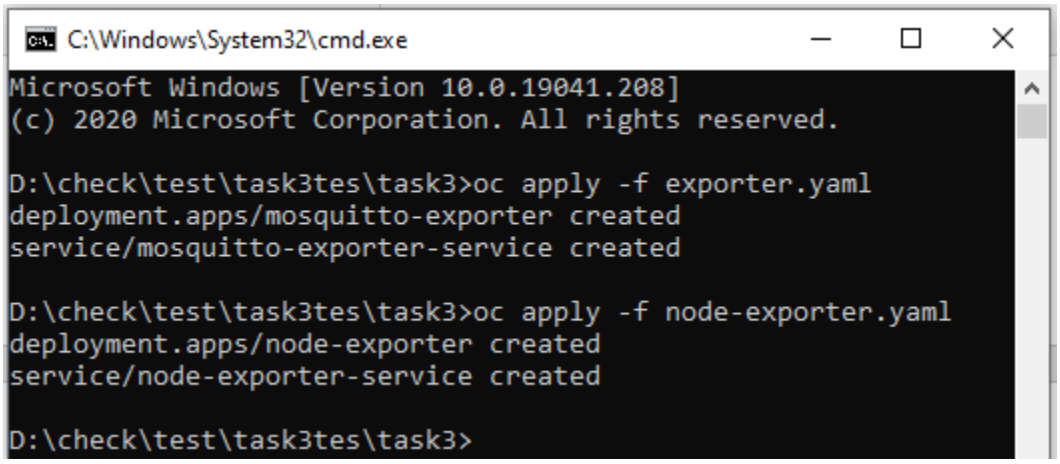
```
C:\Windows\System32\cmd.exe

--> Creating resources ...
    serviceaccount "prometheus-sa" created
    rolebinding.authorization.openshift.io "prometheus-role" created
    deploymentconfig.apps.openshift.io "prometheus" created
    secret "prometheus-nginx-secret" created
    configmap "prometheus-nginx-config" created
    configmap "prometheus-config" created
    persistentvolumeclaim "prometheus-data" created
    service "prometheus-service" created
    route.route.openshift.io "prometheus-route" created
    deploymentconfig.apps.openshift.io "grafana" created
    secret "grafana-secret" created
    configmap "grafana-config" created
    persistentvolumeclaim "grafana-data" created
    service "grafana-service" created
    route.route.openshift.io "grafana-route" created
--> Success
    Access your application via route 'prometheus-route-t3testing.rahtiapp.fi'
    Access your application via route 'grafana-route-t3testing.rahtiapp.fi'
    Run 'oc status' to view your app.

D:\check\test\task3tes\task3>
```

Task 3.2 & 3.3

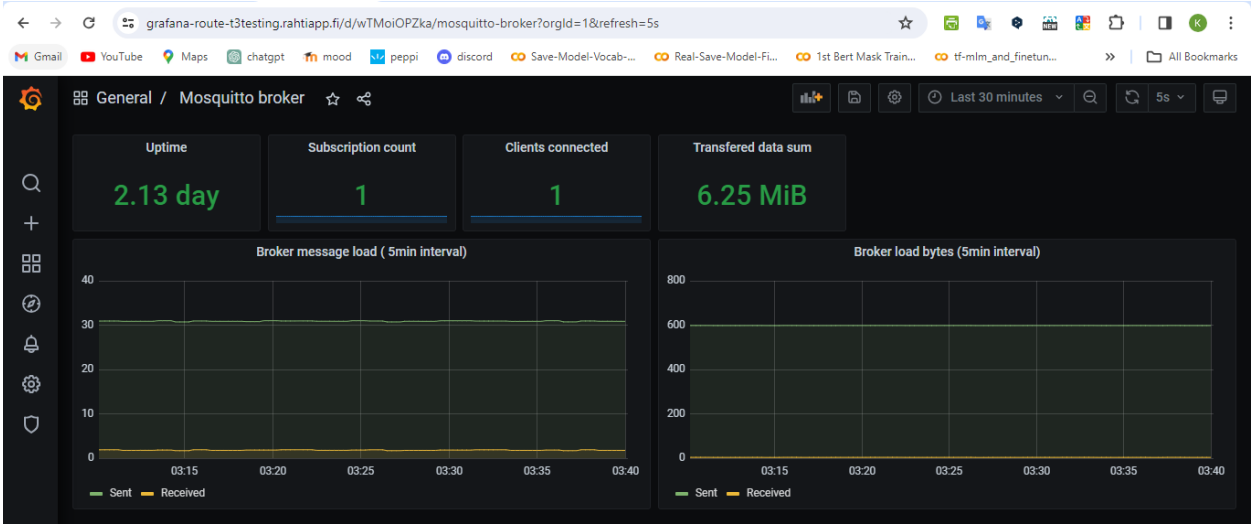
To complete this task 3.2 and 3.3, I have scrapped different metrics by defining different jobs in rahti template.

Then further, I created the deployments and services for mosquito exporter and node exporter:

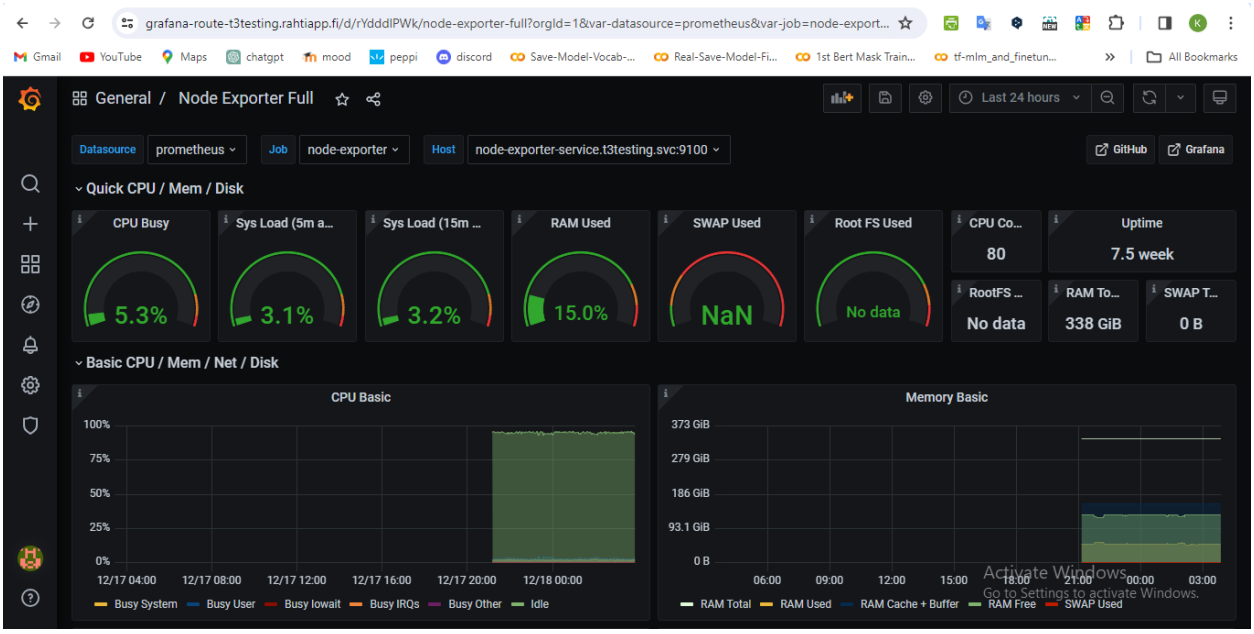
- ✓ oc apply -f exporter.yaml
- ✓ oc apply -f node-exporter.yaml



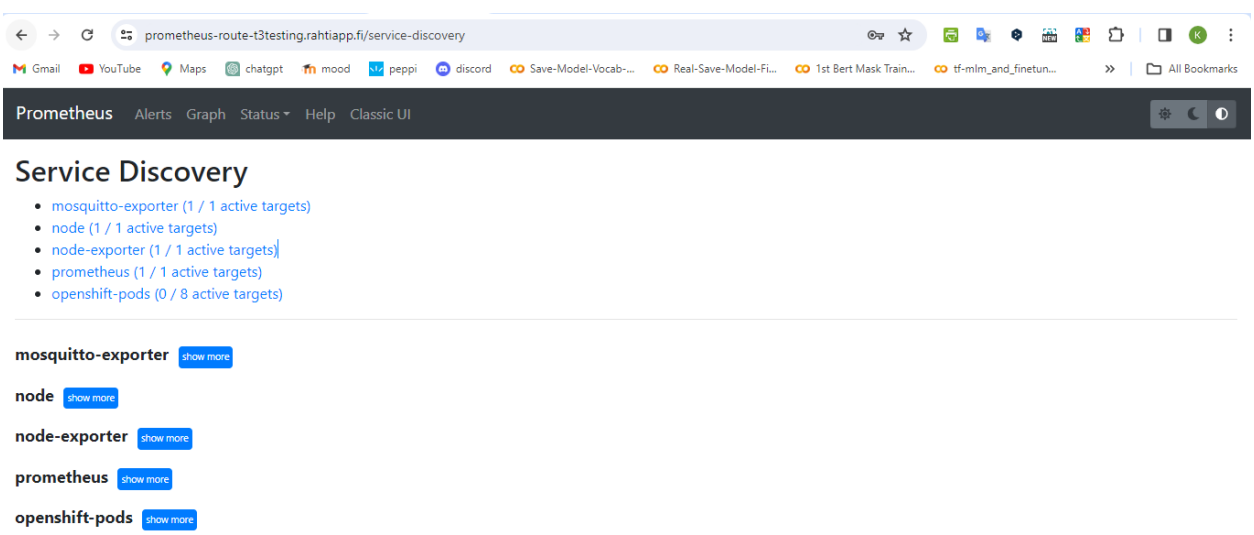
In **task 3.2**, I used the pod which I have created in exercise 3. Through Grafana I was able to check different metrics of Mosquitto Broker such as uptime, traffic, broker load, messages send and received etc on Grafana interface after uploading the Mosquitto Broker Grafana Dashboard. These metrics help the user to measure the performance of mosquitto broker. The Uptime refers to the to the amount of time that a system, service, or application has been available. It represents how many clients are connected to the application etc.:



In task 3.3, I have created a node-exporter pod and then monitor it through the Prometheus and Grafana Dashboard. Below is the screenshot for Node Exporter Full dashboard which I imported from the GitHub repository. It is showing different metrics such as CPU Busy, RAM Used (showing the percentage like how much CPU and RAM have been utilized), Sys Load (it shows the ratio of Load on system) etc.:



Below is a screenshot for Prometheus Dashboard which represents all the jobs with their target, group and job names:



Link for Grafana & Prometheus

Grafana:

- ✓ https://grafana-route-t3testing.rahtiapp.fi/

Link for Mosquitto Broker Dashboard:

- ✓ https://grafana-route-t3testing.rahtiapp.fi/d/wTMoiOPZka/mosquitto-broker?orgId=1&refresh=5s

Link for Node-Exporter Full Dashboard:

- ✓ https://grafana-route-t3testing.rahtiapp.fi/d/rYdddlPWk/node-exporter-full?orgId=1

Prometheus:

- ✓ https://prometheus-route-t3testing.rahtiapp.fi/