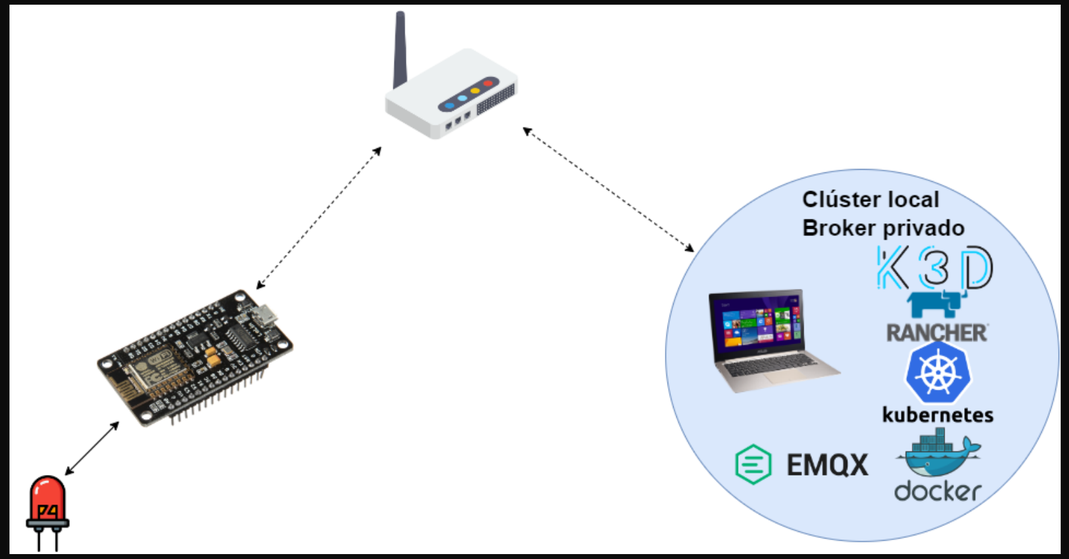
VICTOR MANUEL VELASCO HERNÁNDEZ
JEANETTE MURILLO CORTES

COMPUTACIÓN TOLERANTE A FALLAS
CALENDARIO 2022A
DR. MICHEL EMMANUEL LÓPEZ FRANCO

PROYECTO - IoT

INTRODUCCIÓN

COMPUTACIÓN TOLERANTE



Conexión de una placa NodeMCU8266 a un clúster de Kubernetes para dar una primera etapa a un sistema IoT.

Introducción

TECNOLOGÍAS UTILIZADAS

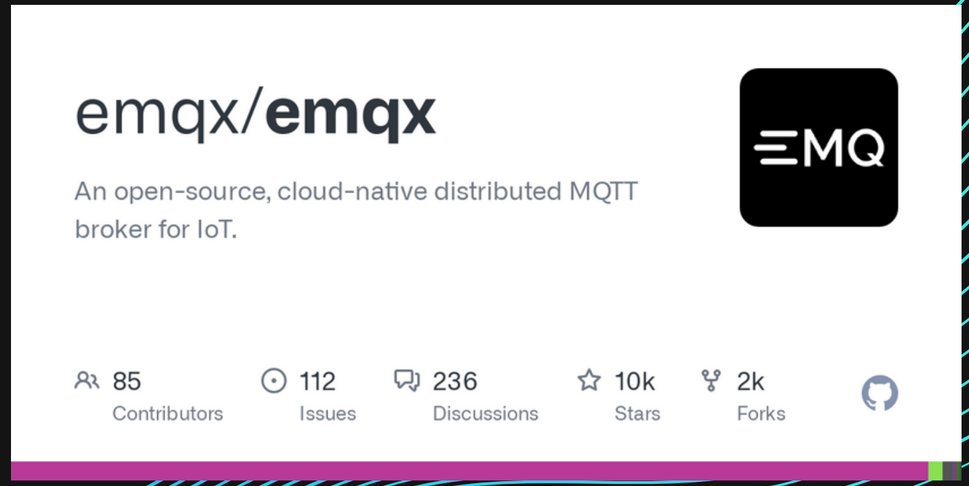
Se expondran las tecnologías que fueron necesarias para realizar el proyecto.

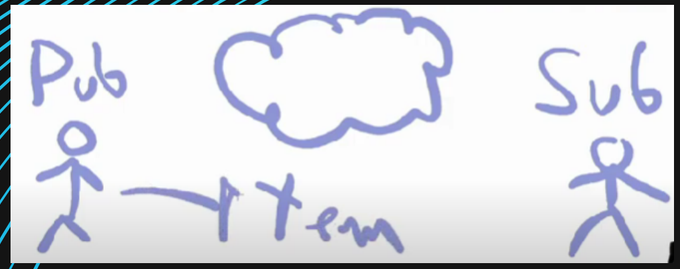
CONEXIÓN NODEMCU8266

Conexión de una placa NodeMCU8266 a un clúster de Kubernetes para dar una primera etapa a un sistema IoT.

EMQX

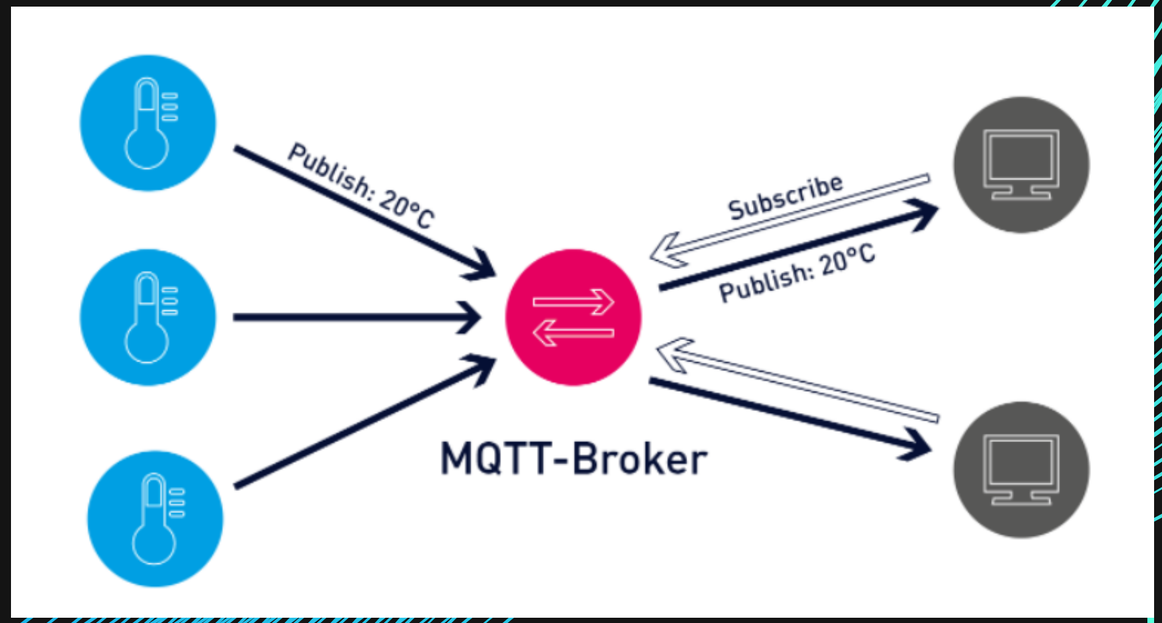
Es un proyecto open-source, apoyado por Cloud Native, el cuál funge como broker MQTT para IoT.





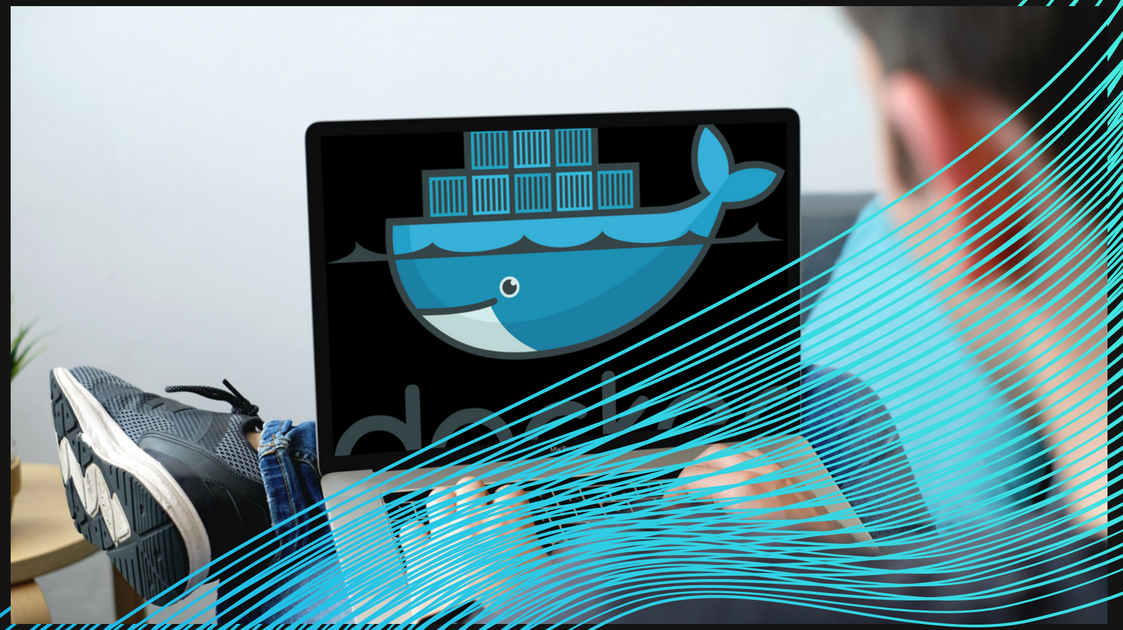
MQTT

MQTT (Message Queuing Telemetry Transport) es un protocolo de comunicación maquina a maquina para redes de bajos recursos.



Docker

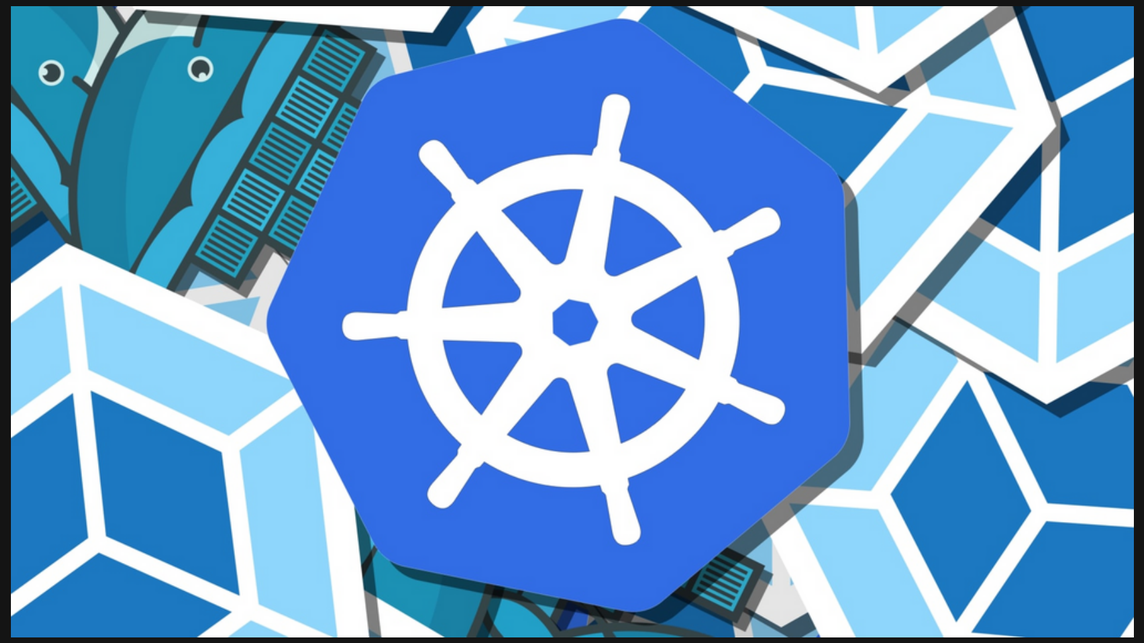
Se hará uso de la imagen "emqx/emqx", con el cuál será posible desplegarlo en Kubernetes.



DOCKER PULL EMQX/EMQX:4.4.3

DOCKER RUN -D --NAME EMQX -P 1883:1883 -P 8081:8081 -P 8083:8083 -P 8084:8084 -P 8883:8883 -P 18083:18083 EMQX/EMQX:4.4.3

Kubernetes



Se creó un Deployment y Service para este clúster.

K3S

K3S es una versión ligera de Kubernetes, el cuál trabaja con Rancher (K3D).

Rancher

Rancher como gestor del clúster.

NodeMCU8266

Placa de desarrollo que cuenta con el módulo ESP8266, el cuál contiene un dispositivo Wifi de 2.4Ghz.

Instalación

Es fundamental tener instalado Docker, y para facilitar los procesos de intalación se instaló un Packet Manager llamado "Chocolatey" para instalar los programas faltantes.


```
CHOCO INSTALL K3D -Y
>> CHOCO INSTALL JQ -Y
>> CHOCO INSTALL YQ -Y
>> CHOCO INSTALL KUBERNETES-HELM -Y
```

Configuración de K3D

Se creó un cluster "IoT" en K3D, con su respectivo comando, declarando los puertos que serán expuestos:

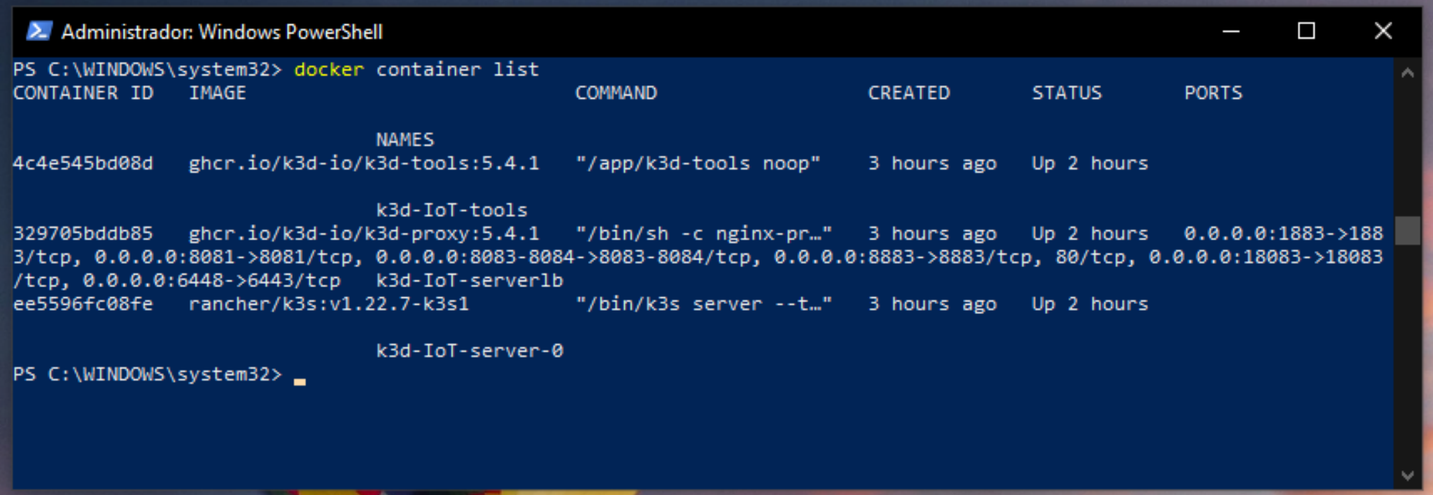
```
K3D CLUSTER CREATE --API-PORT 6448 -P
"18083:18083" -P "1883:1883" -P "8883:8883" -P
"8081:8081" -P "8083:8083" -P "8084:8084"  IOT
```

Por otro lado, es importante configurar la variable de entorno dentro del sistema, en este caso es Windows 10 con WSL:

```
K3D KUBECONFIG GET ${ENV:CLUSTER_NAME}
> $ENV:KUBECONFIG_FILE

$ENV:KUBECONFIG=($ENV:KUBECONFIG_FILE)
```

Configuración de K3D



Configuración de Deployment y Service

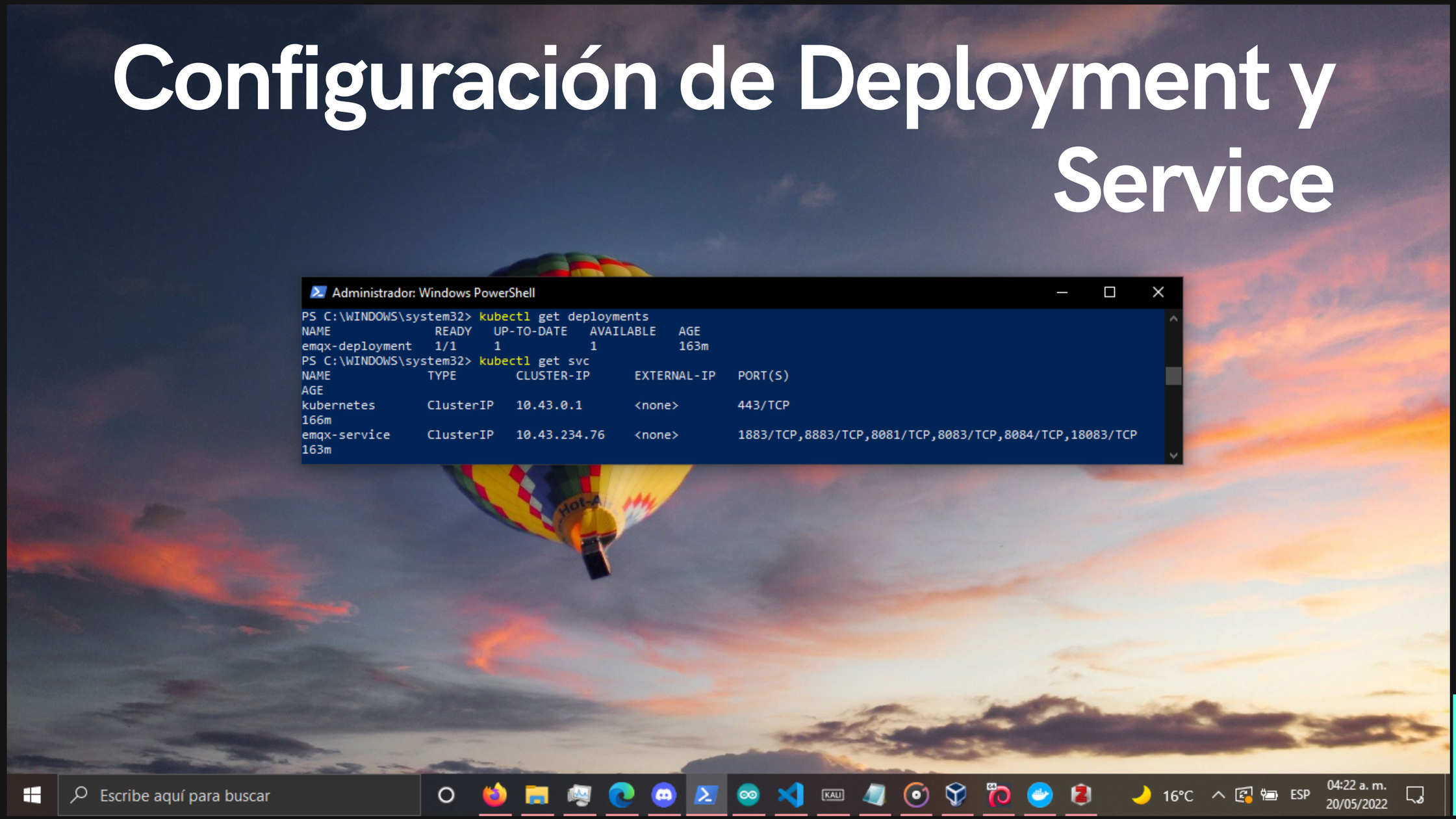
Se hará un Deployment y Servicio dentro de K3S que use la imagen "emqx/emqx", asignando los puertos correspondientes a cada uno de los servicios internos, todo esto lo lograremos con un archivo YAML generado.

K3S KUBECTL APPLY -F EMQX.YAML

EMQX.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: emqx-deployment
  labels:
    app: emqx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: emqx
  template:
    metadata:
      labels:
        app: emqx
    spec:
      containers:
      - name: emqx
        image: emqx/emqx:v4.1-rc.1
        ports:
        - name: mqtt
          containerPort: 1883
        - name: mqttssl
          containerPort: 8883
        - name: mgmt
          containerPort: 8081
```

Configuración de Deployment y Service



Creando servicios

Se crearan servicios para cada servicio interno del despliegue, para lo cuál se usaran de tipo NodePort para poderlos exponer al exterior, a continuación, ejemplos:

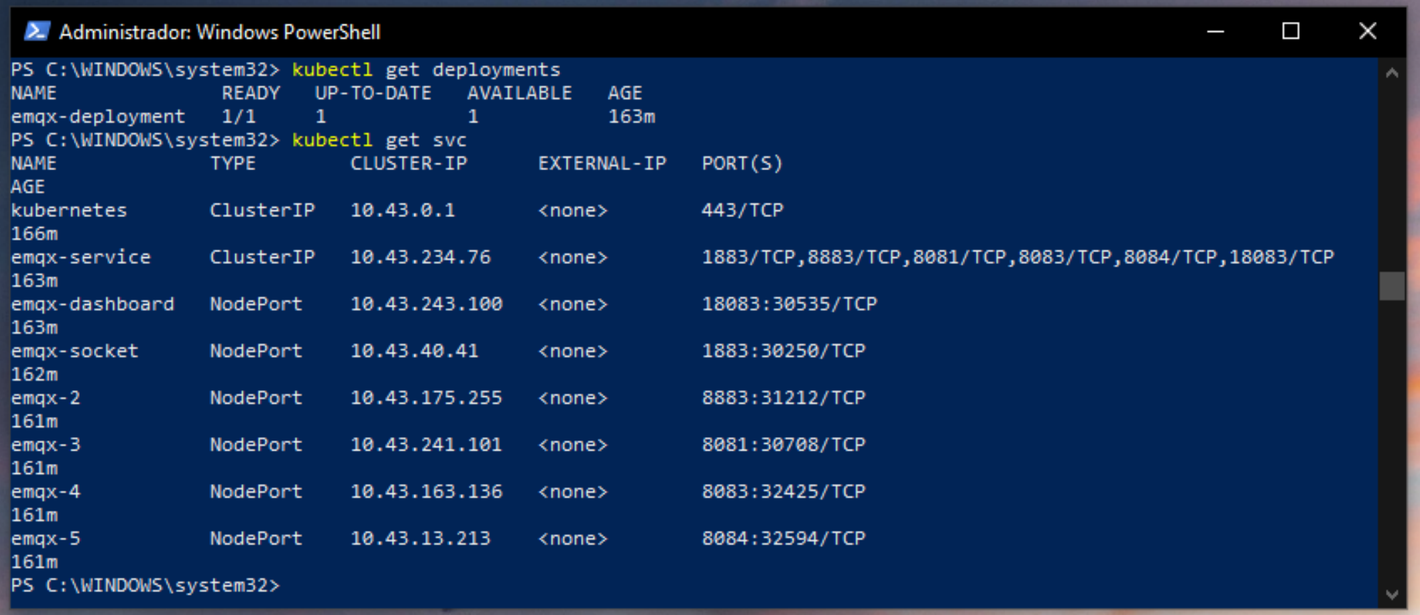
```
KUBECTL EXPOSE DEPLOYMENT EMQX-DEPLOYMENT --NAME EMQX-DASHBOARD --PORT 18083 --TARGET-PORT=18083 --TYPE NODEPORT

KUBECTL EXPOSE DEPLOYMENT EMQX-DEPLOYMENT --NAME EMQX-SOCKET --PORT 1883 --TARGET-PORT=1883 --TYPE NODEPORT
```

EMQX.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: emqx-deployment
  labels:
    app: emqx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: emqx
  template:
    metadata:
      labels:
        app: emqx
    spec:
      containers:
      - name: emqx
        image: emqx/emqx:v4.1-rc.1
        ports:
        - name: mqtt
          containerPort: 1883
        - name: mqttssl
          containerPort: 8883
        - name: mgmt
          containerPort: 8081
```

Creando servicios



Conectando puertos

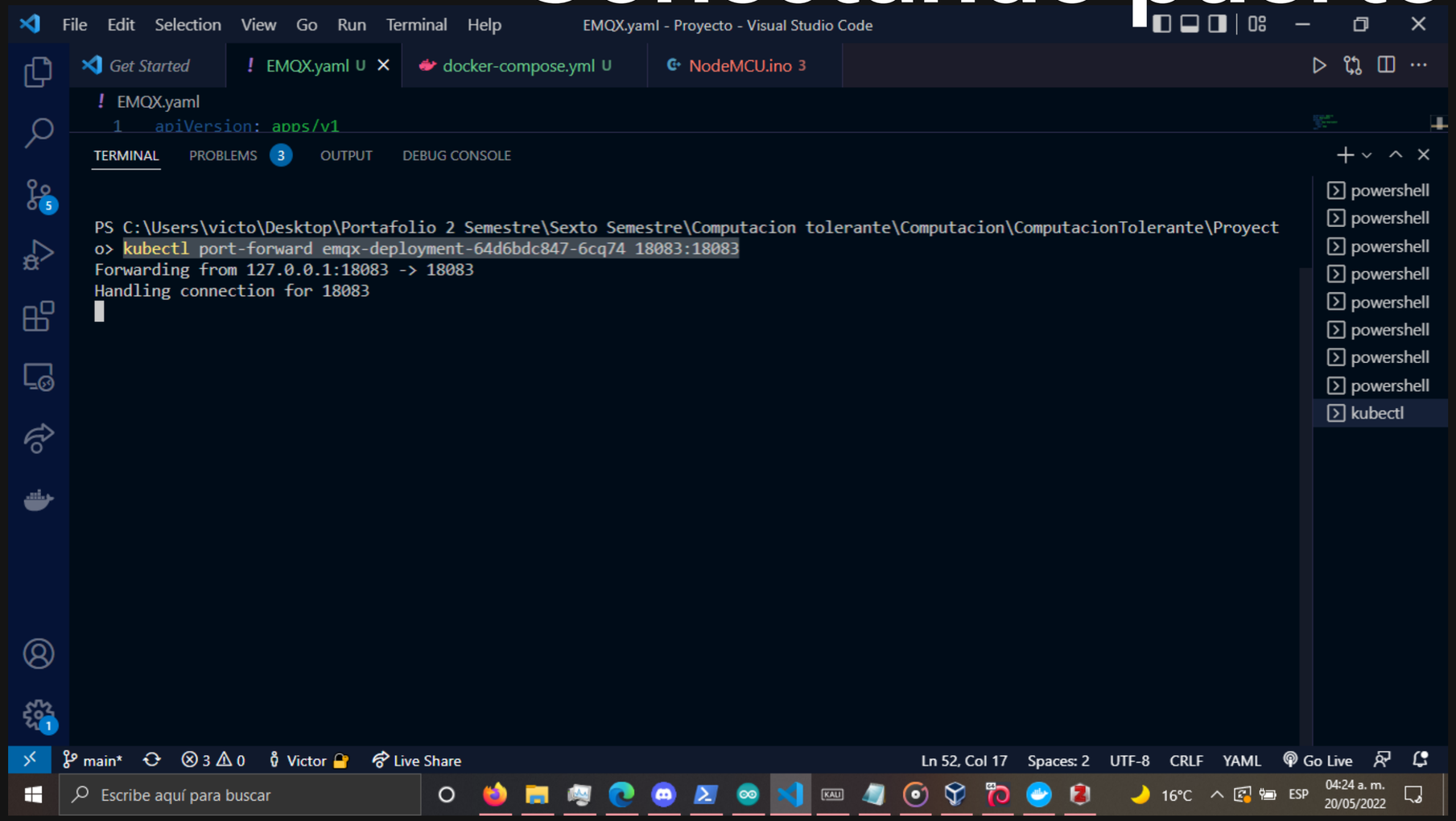
Y por ultimo los servicios los enlazamos con los puertos del servidor, a continuación, ejemplos de lo que se implementará:

KUBECTL PORT-FORWARD EMQX-DASHBOARD 18083:18083

EMQX.yaml

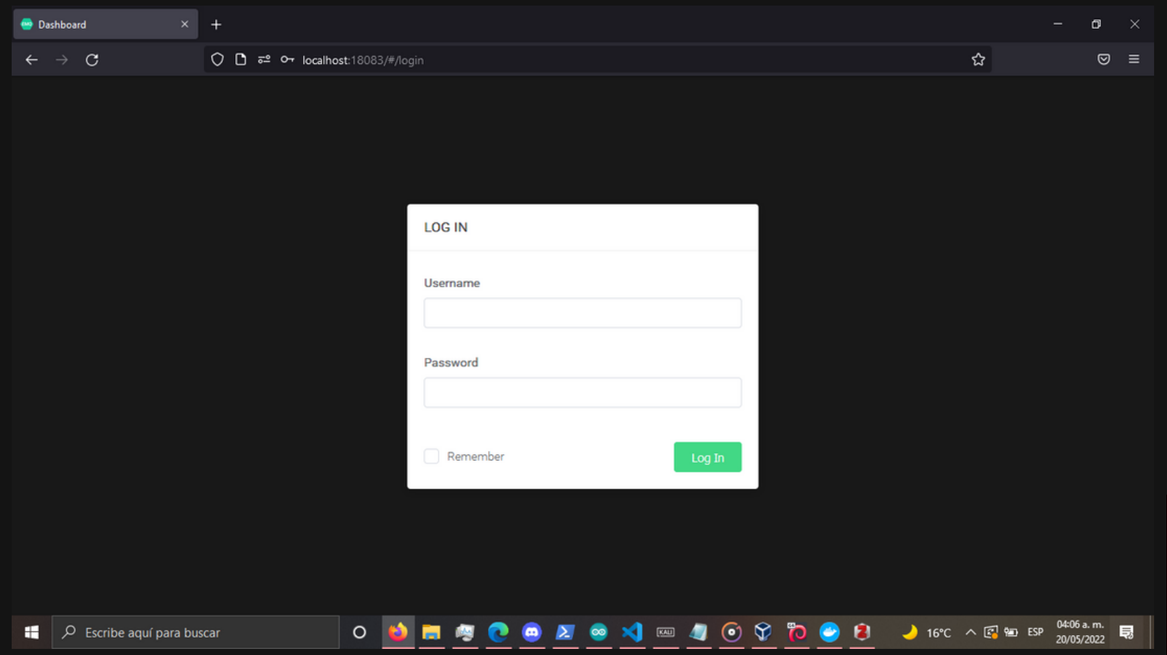
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: emqx-deployment
  labels:
    app: emqx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: emqx
  template:
    metadata:
      labels:
        app: emqx
    spec:
      containers:
      - name: emqx
        image: emqx/emqx:v4.1-rc.1
        ports:
        - name: mqtt
          containerPort: 1883
        - name: mqttssl
          containerPort: 8883
        - name: mgmt
          containerPort: 8081
```

Conectando puertos

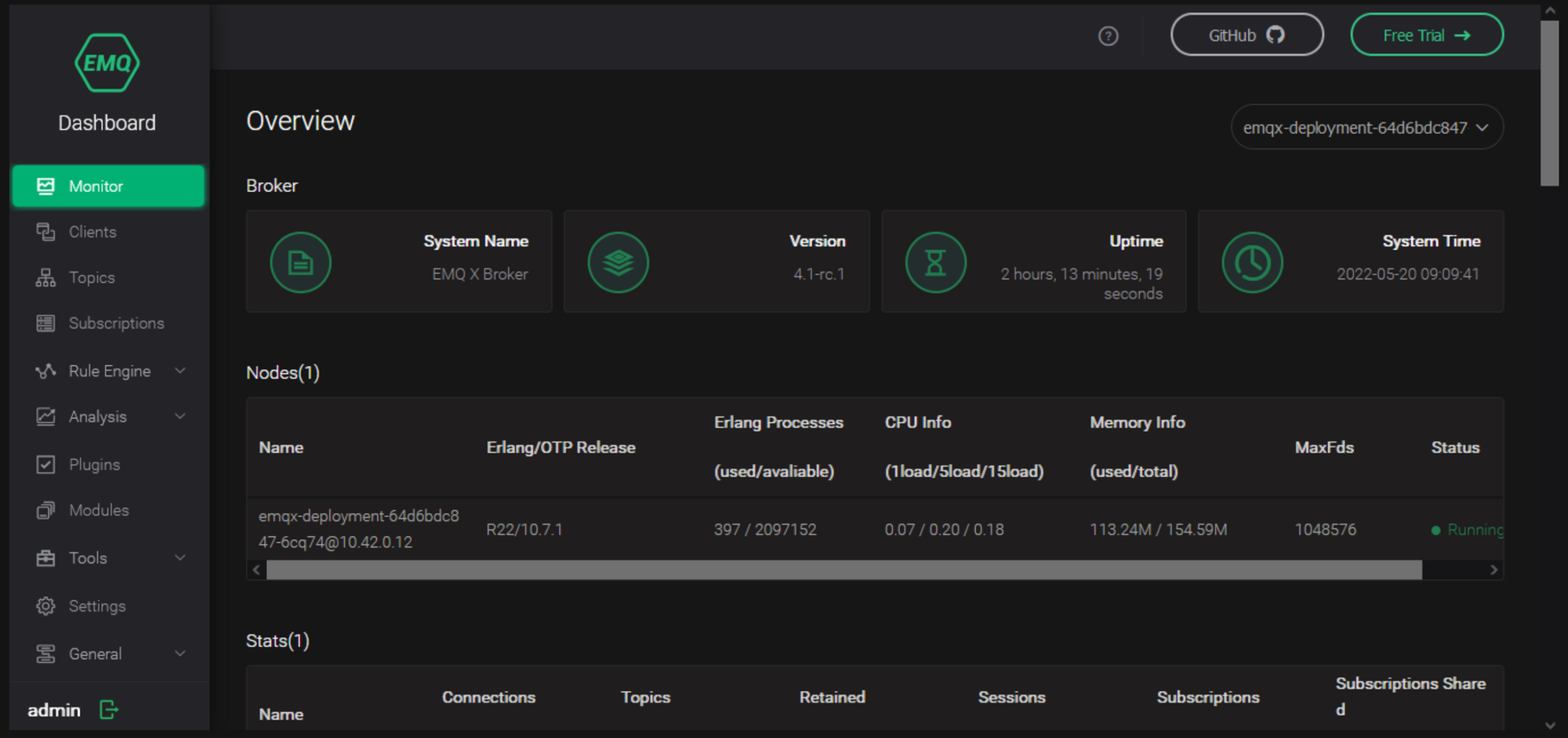


Dashboard

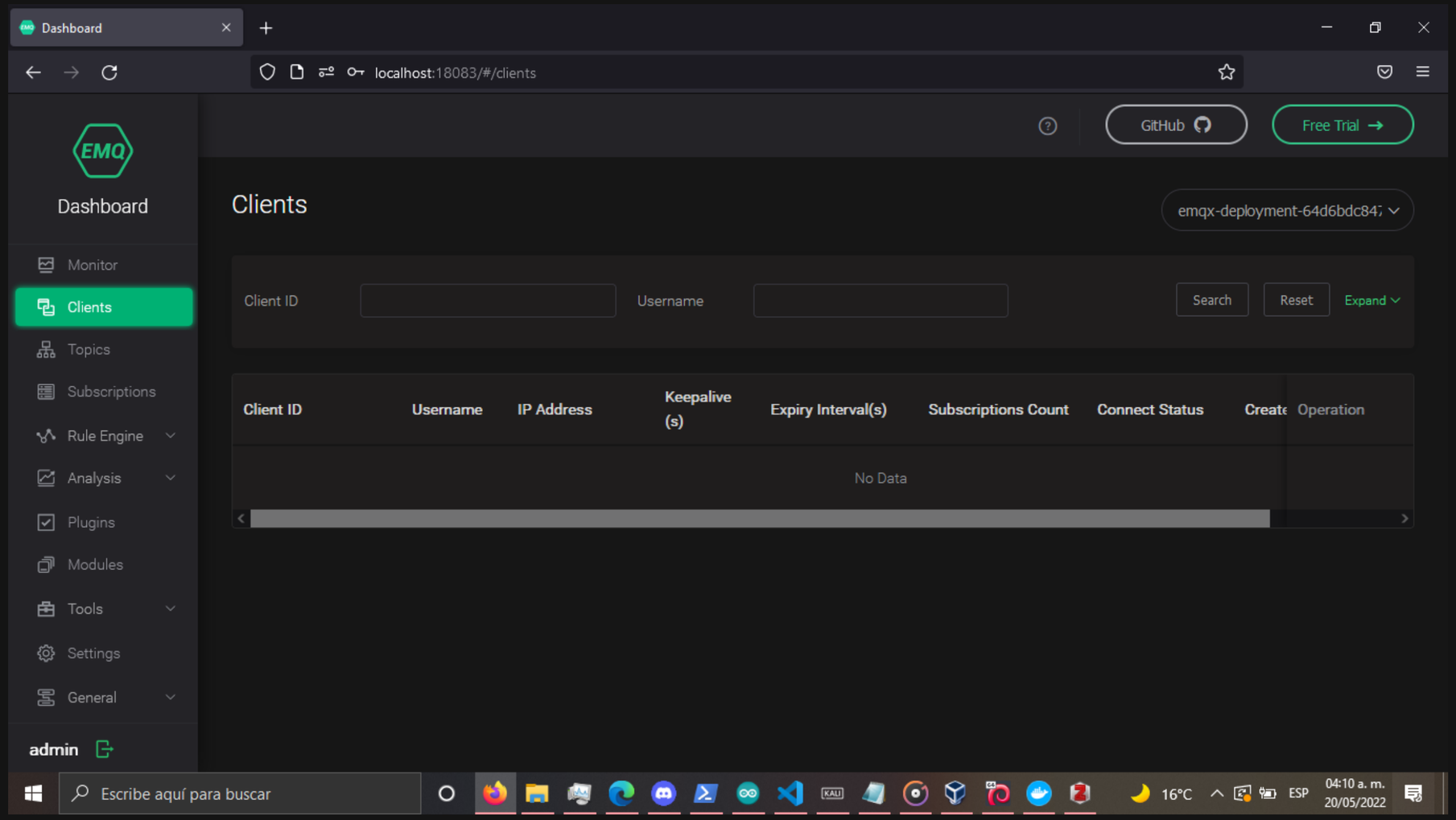
Al haber sincronizado los puertos, nos metemos al Dashboard, con puerto 18083, en el cuál tendremos que logearnos por primera vez, usaremos el usuario default.
Usuario: admin
Password: public



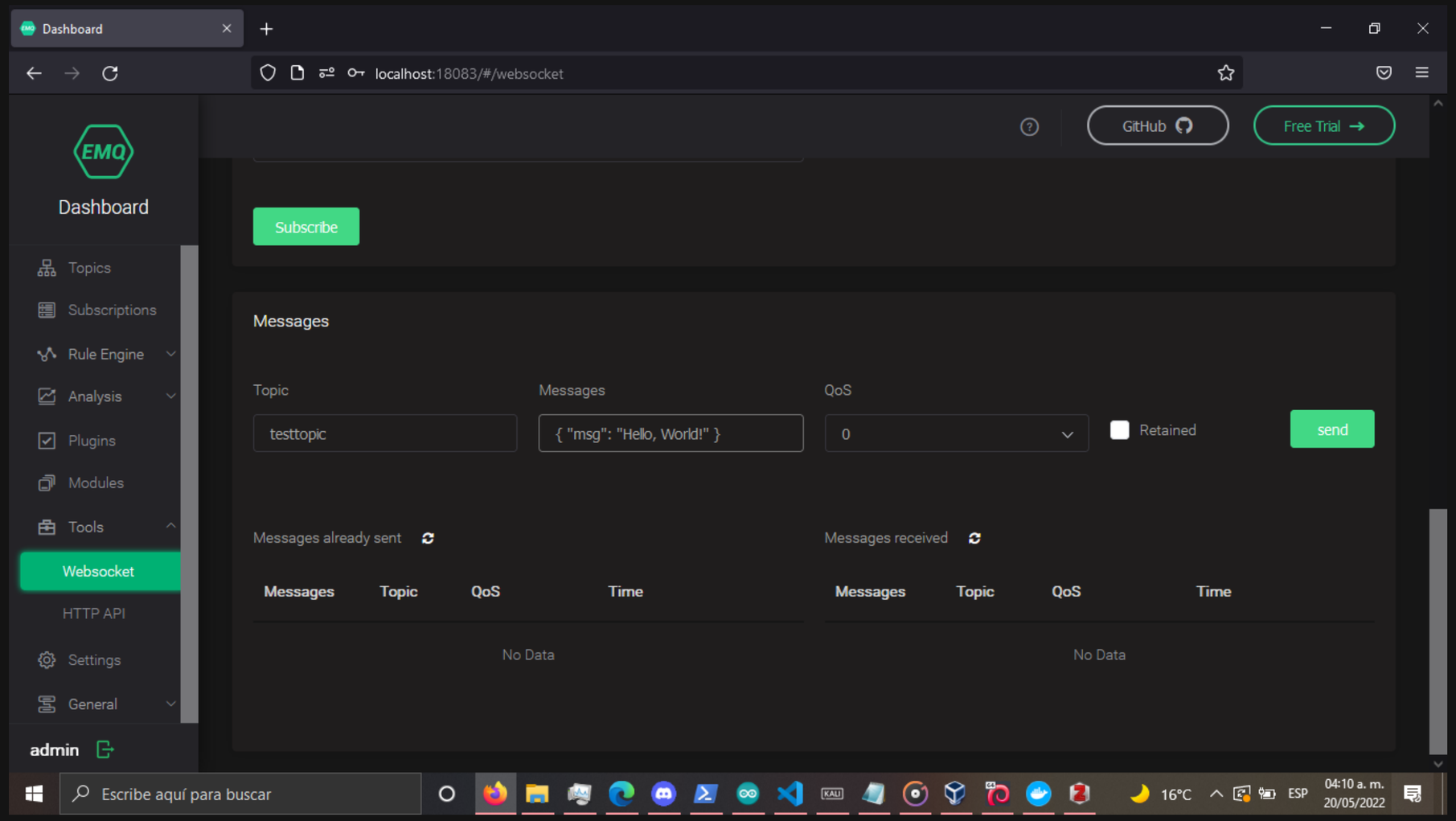
Dashboard


EMQ
Dashboard
Monitor
Clients
Topics
Subscriptions
Rule Engine
Analysis
Plugins
Modules
Tools
Settings
General
admin
GitHub
Free Trial
Overview
emqx-deployment-64d6bdc847
Broker
System Name
EMQ X Broker
Version
4.1-rc.1
Uptime
2 hours, 13 minutes, 19 seconds
System Time
2022-05-20 09:09:41
Nodes(1)
Name
Erlang/OTP Release
Erlang Processes (used/avaliable)
CPU Info (1load/5load/15load)
Memory Info (used/total)
MaxFds
Status
emqx-deployment-64d6bdc847-6cq74@10.42.0.12
R22/10.7.1
397 / 2097152
0.07 / 0.20 / 0.18
113.24M / 154.59M
1048576
Running
Stats(1)
Name
Connections
Topics
Retained
Sessions
Subscriptions
Subscriptions Shared

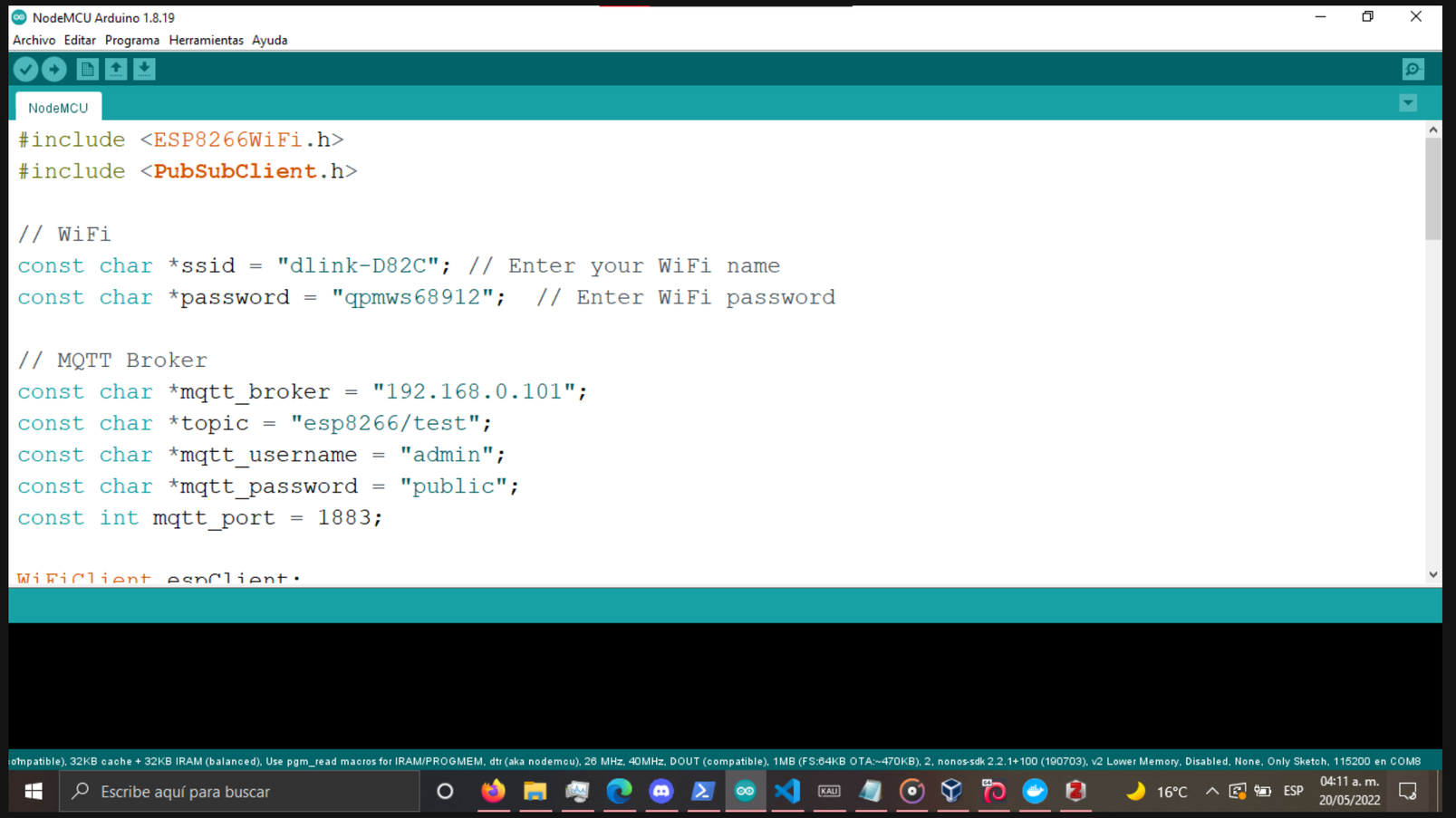
Dashboard


Dashboard
localhost:18083/#/clients
GitHub
Free Trial
EMQ
Dashboard
Monitor
Clients
Topics
Subscriptions
Rule Engine
Analysis
Plugins
Modules
Tools
Settings
General
admin
Clients
emqx-deployment-64d6bdc847
Client ID
Username
Search
Reset
Expand
Client ID
Username
IP Address
Keepalive (s)
Expiry Interval(s)
Subscriptions Count
Connect Status
Create
Operation
No Data
Escribe aquí para buscar
16°C
ESP
04:10 a. m.
20/05/2022

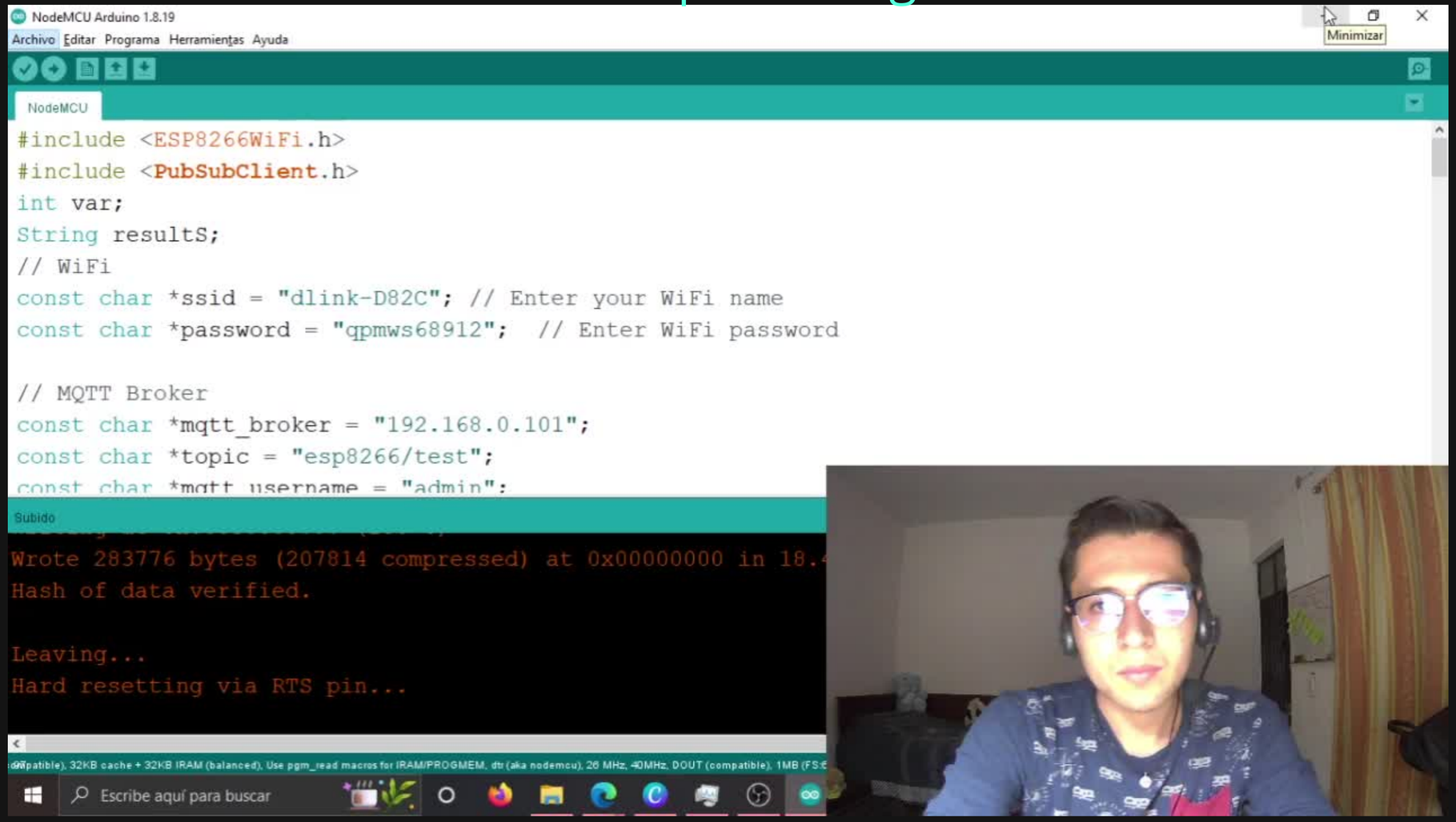
Dashboard


Dashboard
localhost:18083/#/websocket
GitHub
Free Trial
EMQ
Dashboard
Topics
Subscriptions
Rule Engine
Analysis
Plugins
Modules
Tools
Websocket
HTTP API
Settings
General
admin
Subscribe
Messages
Topic
testtopic
Messages
{ "msg": "Hello, World!" }
QoS
0
Retained
send
Messages already sent
Messages received
Messages
Topic
QoS
Time
No Data
Escribe aquí para buscar
16°C
ESP
04:10 a. m.
20/05/2022

NodeMCU8266



Link de Drive:
https://drive.google.com/file/d/1v7-_CIrggtQCW6O1r4HuX6lM3AB7Bfwz/view?usp=sharing



Failure Injection

Se hará uso de la herramienta de Chaos Engineering, Litmus, es open-source, proyecto encubado de la CNCF (Cloud Native Cloud Foundation), con este haremos pruebas de "pod-delete" .

Failure Injection

SE INSTALA LITMUS EN EL CLÚSTER:

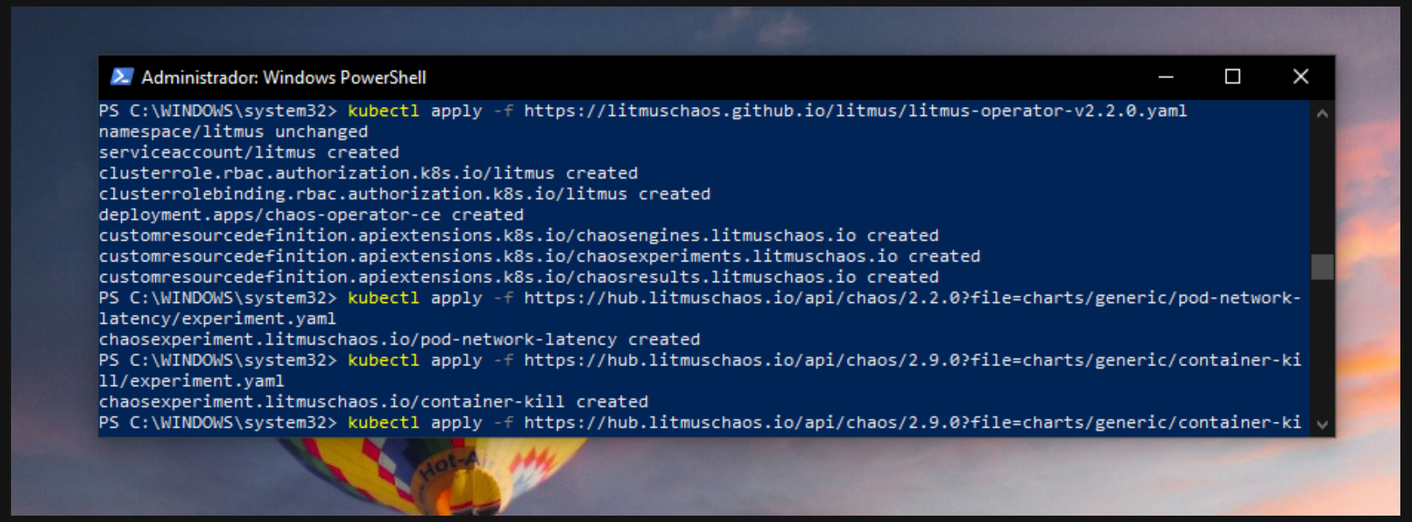
KUBECTL APPLY -F HTTPS://LITMUSCHAOS.GITHUB.IO/LITMUS/LITMUS-OPERATOR-V2.2.0.YAML

SE INSTALA EL CHAOS EXPERIMENT "POD-DELETE":

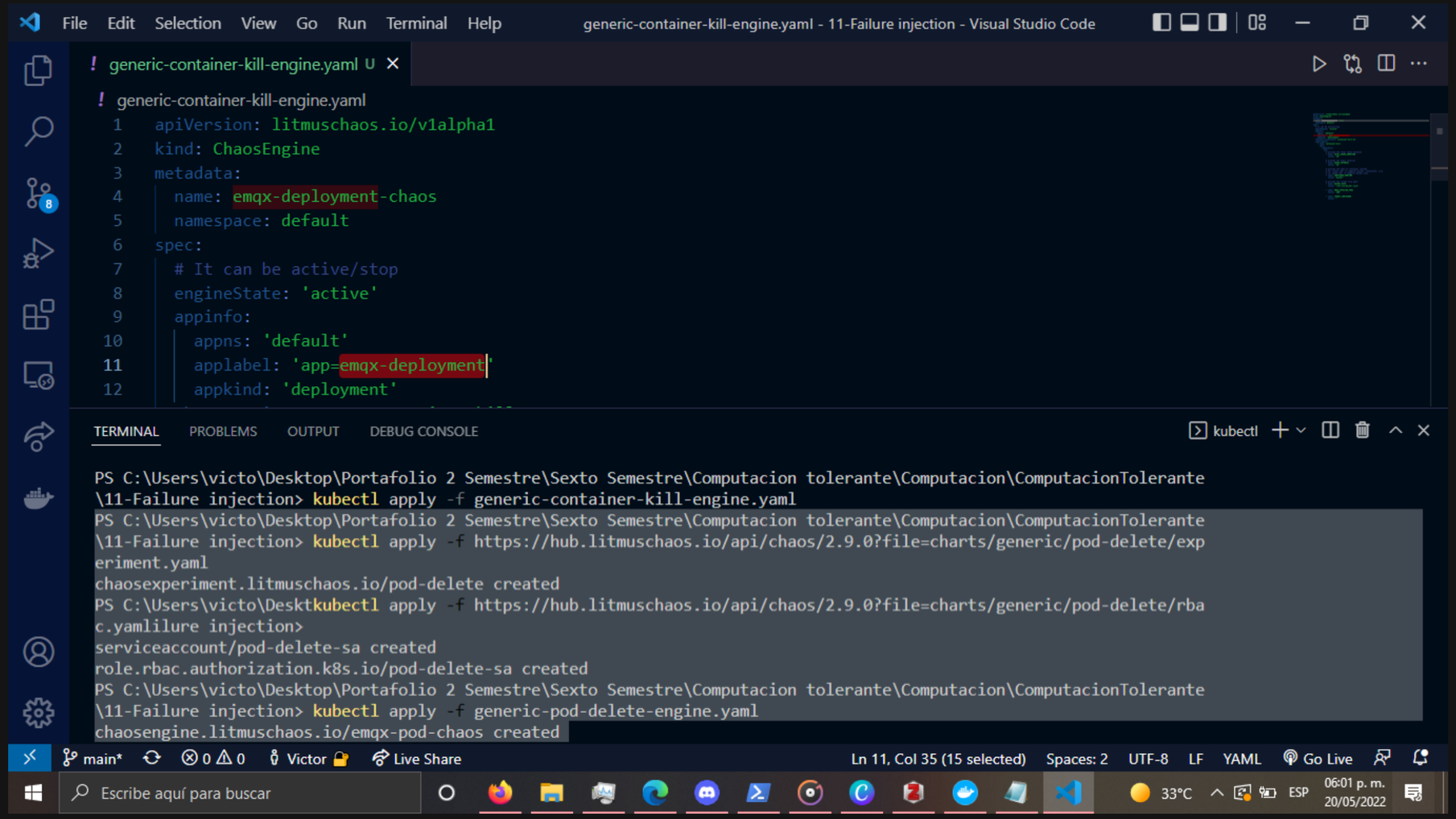
KUBECTL APPLY -F HTTPS://HUB.LITMUSCHAOS.IO/API/CHAOS/2.9.0?FILE=CHARTS/GENERIC/POD-DELETE/EXPERIMENT.YAML

SE CONCEDEN PERMISOS(RBAC) PARA QUE EL EXPERIMENTO TRABAJE:

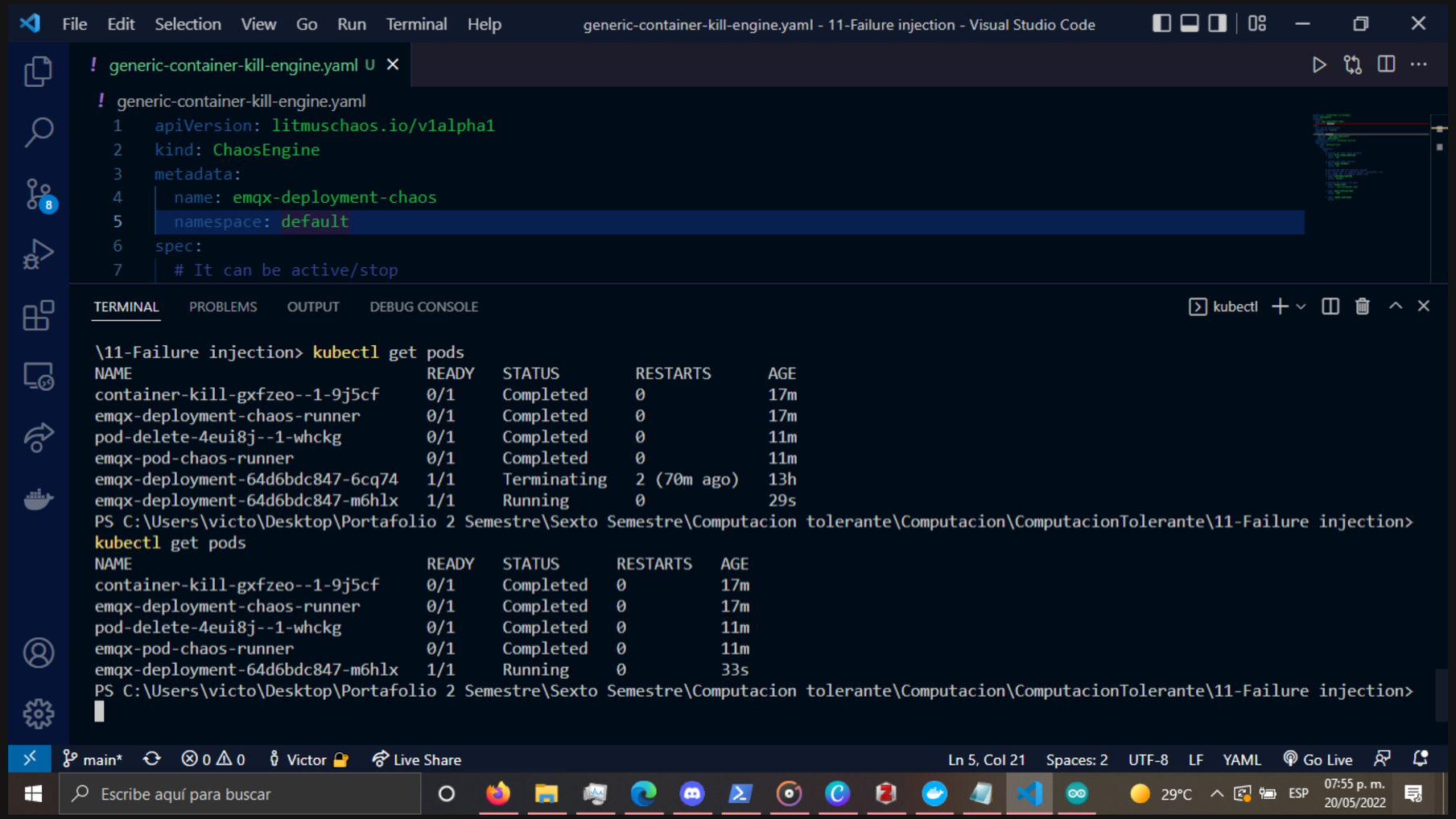
KUBECTL APPLY -F HTTPS://HUB.LITMUSCHAOS.IO/API/CHAOS/2.9.0?FILE=CHARTS/GENERIC/POD-DELETE/RBAC.YAML



Después, descargaremos el archivo YAML que incluye el sitio web del "pod-deletion", para ajustar la configuración y después implementarla en el clúster.



Podemos apreciar que el pod ha sido eliminado, y se ha replicado exitosamente, por lo tanto es tolerante a "pod-deletion"

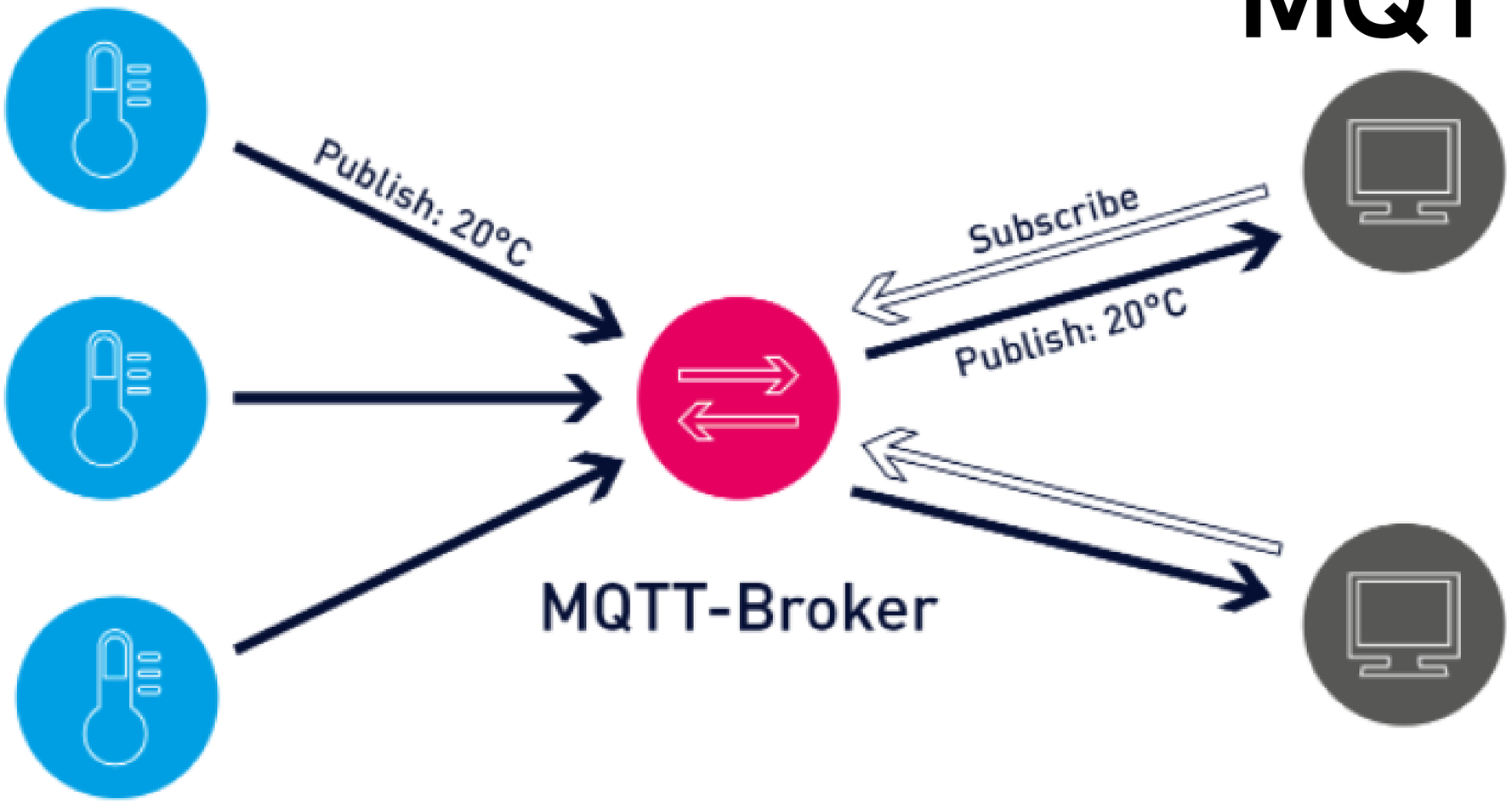


Reto

Se ha propuesto el reto de conectar más de una placa de desarrollo con las cuales:

- Se prenderán sus focos con un solo mensaje.
- Uso de sensores para enviar mensaje.

MQTT

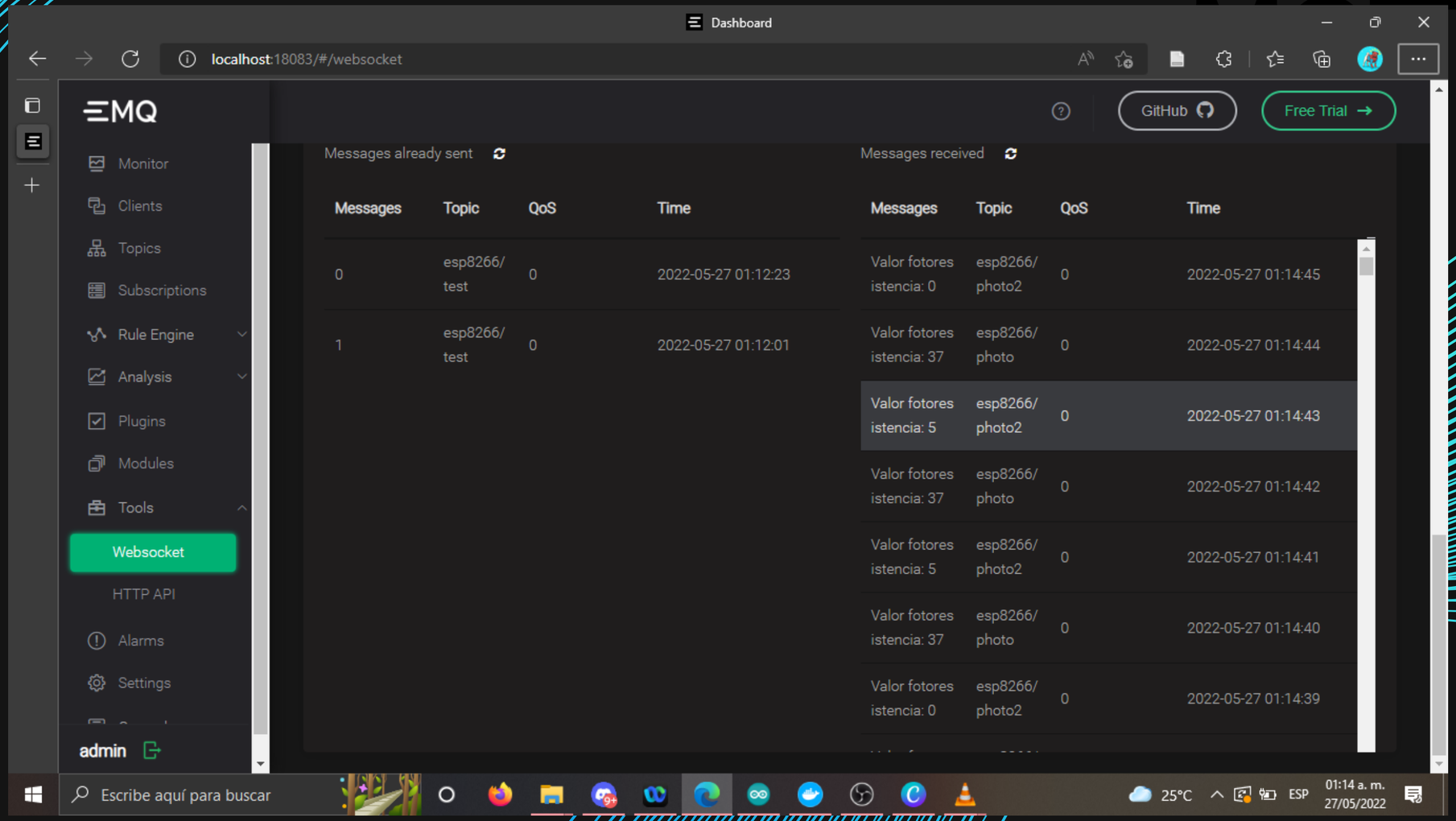


LINK DE DRIVE:
HTTPS://DRIVE.GOOGLE.COM/FILE/D/1DDA4RBIDVZXG3WM8QDMOKXUMUQB20X7B/VIEW?USP=SHARING

Reto

Se ha propuesto el reto de conectar más de una placa de desarrollo con las cuales:

- Se prenderán sus focos con un solo mensaje.
- Uso de sensores para enviar mensaje.


Dashboard
localhost:18083/#/websocket
EMQ
GitHub
Free Trial
Monitor
Clients
Topics
Subscriptions
Rule Engine
Analysis
Plugins
Modules
Tools
Websocket
HTTP API
Alarms
Settings
admin
Messages already sent
Messages received
Messages Topic QoS Time
0 esp8266/test 0 2022-05-27 01:12:23
1 esp8266/test 0 2022-05-27 01:12:01
Valor fotoresistencia: 0 esp8266/photo2 0 2022-05-27 01:14:45
Valor fotoresistencia: 37 esp8266/photo 0 2022-05-27 01:14:44
Valor fotoresistencia: 5 esp8266/photo2 0 2022-05-27 01:14:43
Valor fotoresistencia: 37 esp8266/photo 0 2022-05-27 01:14:42
Valor fotoresistencia: 5 esp8266/photo2 0 2022-05-27 01:14:41
Valor fotoresistencia: 37 esp8266/photo 0 2022-05-27 01:14:40
Valor fotoresistencia: 0 esp8266/photo2 0 2022-05-27 01:14:39
Escribe aquí para buscar
25°C
ESP
01:14 a. m.
27/05/2022

GRACIAS POR SU ATENCIÓN