

Πλατφόρμες Cloud

Εργασία Μεταπτυχιακού

ΕΛΕΥΘΕΡΙΟΣ ΣΤΕΒΗΣ 24118

Περιεχόμενα

Τεχνολογίες αρχιτεκτονική	2
API	3
Node-red.....	4
Minio	8
RABBITMQ	11
Keycloak.	13
Docker and Kubernetes	17

Στόχος:

Ο στόχος της εργασίας είναι να ταξινομήσουμε τα δεδομένα που προέρχονται από έναν αισθητήρα (sensor), οι οποίοι αφορούν την ποιότητα του αέρα, πιο συγκεκριμένα, τα δεδομένα PM10. Ο sensor βρίσκεται στην Ινδία.

Το PM10 αναφέρεται σε μικροσκοπικά σωματίδια που υπάρχουν στον αέρα και έχουν διάμετρο ίσο ή μικρότερο από 10 μm . Αυτά τα σωματίδια μπορούν να προέρχονται από φυσικές πηγές, όπως τα θραύσματα γης ή το κονιό, ή από ανθρώπινες δραστηριότητες, όπως την καύση καυσίμων ή τις εκπομπές από τις μεταφορές. Τα PM10 θεωρούνται επικίνδυνα για την υγεία, καθώς μπορούν να εισχωρήσουν στον πνευμονικό σύστημα.

Τεχνολογίες αρχιτεκτονική

Οι τεχνολογίες που χρησιμοποιώ για εργασία είναι οι παρακάτω

- Node-red
- RabbitMQ
- Docker +Kubernetes
- Minio
- Keycloak

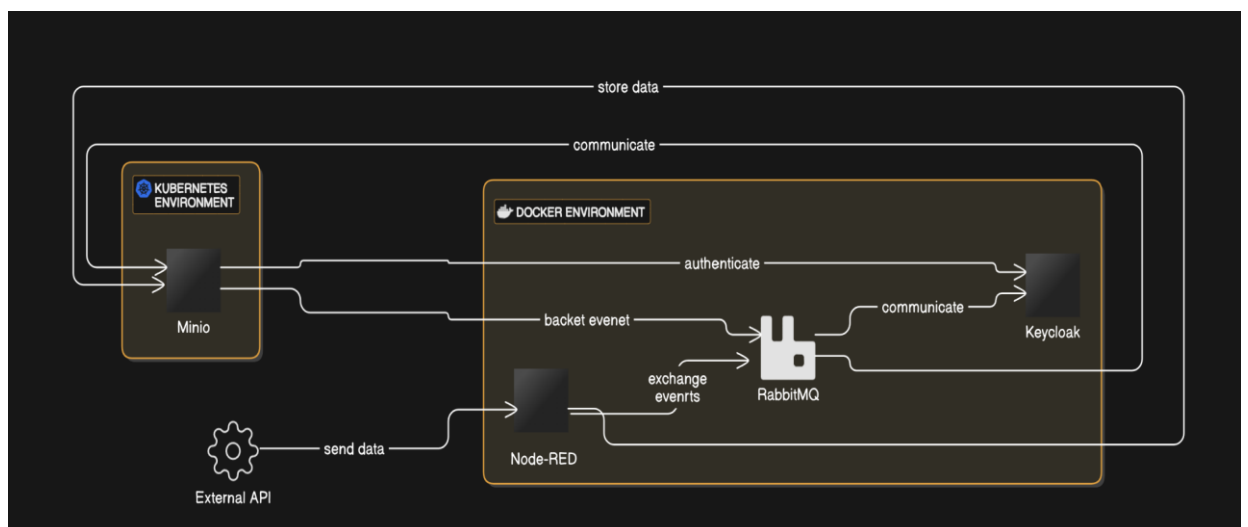


Figure 1 diagram

API

Για να αντλήσω τα δεδομένα για την εργασία, χρησιμοποίησα την <https://openaq.org/>. Έκανα εγγραφή και μου δόθηκε ένα API key, ώστε να μπορεί να πραγματοποιηθεί η ταυτοποίηση του χρήστη που καλεί το API.

Επειδή η εργασία ολοκληρώθηκε πριν από τις 31/01, το API της ιστοσελίδας είχε διαφορετική μορφή και μεγαλύτερη ευελιξία στις κλήσεις. Ωστόσο, από την 01/02 το API αναβαθμίστηκε σε νέα έκδοση με περισσότερους περιορισμούς, γεγονός που έφερε αλλαγές και στην εργασία.

Το call που κάνω είναι αυτό

<https://api.openaq.org/v3/sensors/395/hours/hourofday>

που μου φέρνει για αυτόν το sensor όλες τις τιμές που μπορεί να παίρνει από το περιβάλλον(SO₂,PM₁₀).

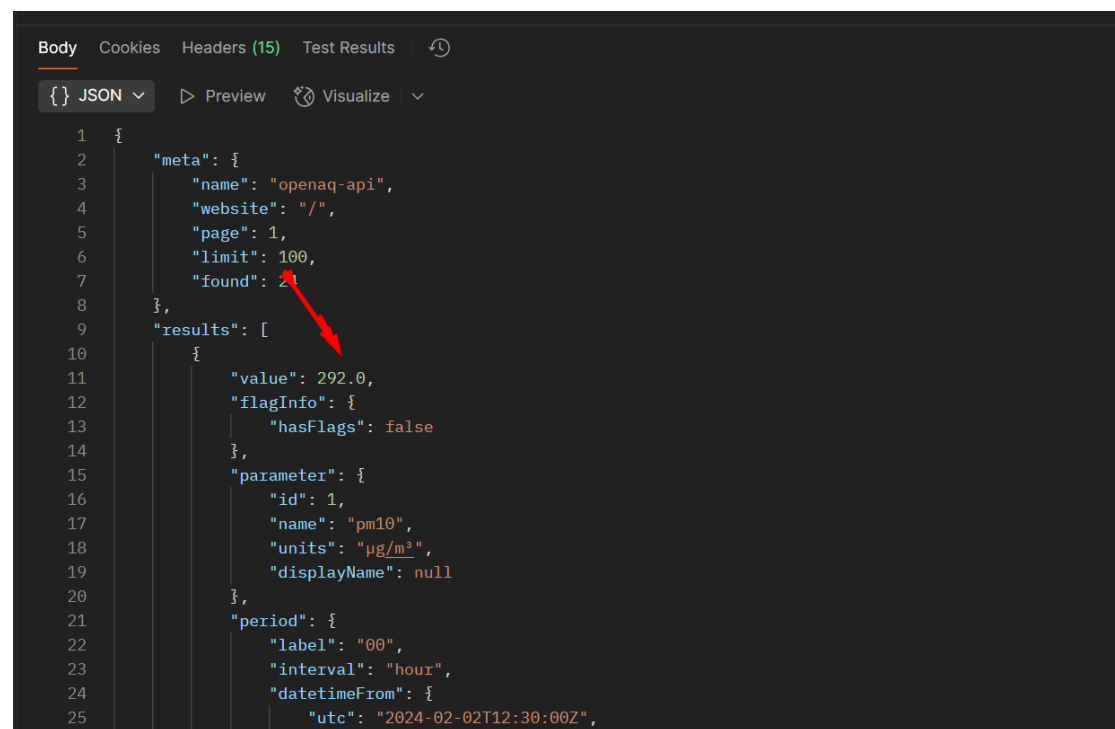


Figure 2 API CALL

Node-red.

Το Node-RED τρέχει σε Docker και το χρησιμοποιώ για τη λήψη δεδομένων από το API και την αποθήκευσή τους στο Minio. Πιο αναλυτικά:

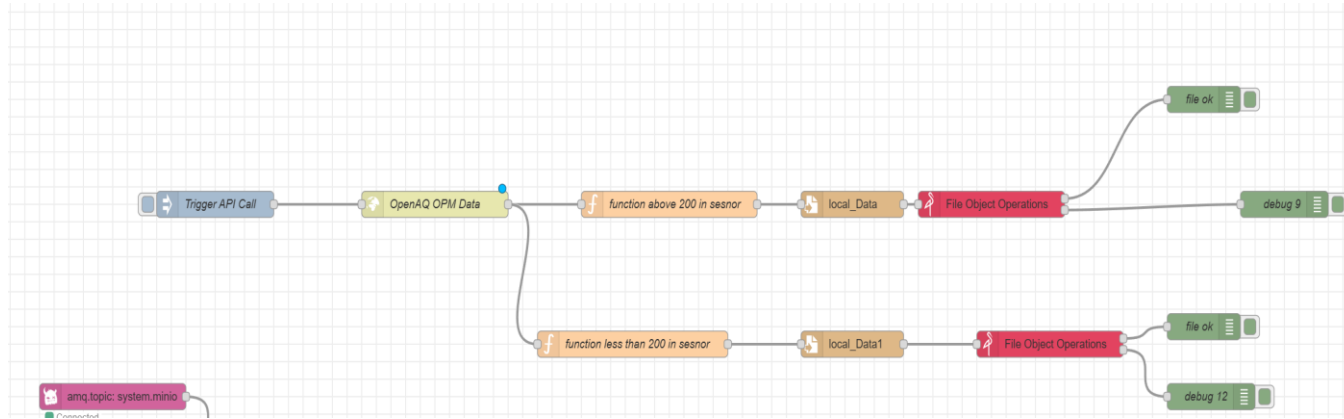


Figure 3 node-red-api-call

Στην παραπάνω φωτογραφία φαίνεται το trigger API call που χρησιμοποιώ για να ξεκινήσει η διαδικασία. Στη συνέχεια, μια function εξάγει μόνο τα απαραίτητα δεδομένα από το JSON αρχείο του API call. Υπάρχει διαχωρισμός ανάλογα με το αν η τιμή είναι πάνω ή κάτω από 200. Έπειτα, τα δεδομένα αποθηκεύονται πρώτα στο local_Data και στη συνέχεια στο Minio. Τέλος, εμφανίζεται ένα output μήνυμα επιβεβαίωσης ότι η διαδικασία ολοκληρώθηκε επιτυχώς.

Το instance του Minio έχει τα παρακάτω ώστε να γίνει η σύνδεση

The image shows the configuration form for a MinIO instance in Node-RED. The form is titled 'Properties' and contains the following fields:

- Name: MinIO Instance
- MinIO Host: host.docker.internal
- Port: 9000
- Access Key: minioadmin
- Secret Key: (masked with dots)
- Use SSL: (unchecked checkbox)

Figure 4 instance minio in node-red

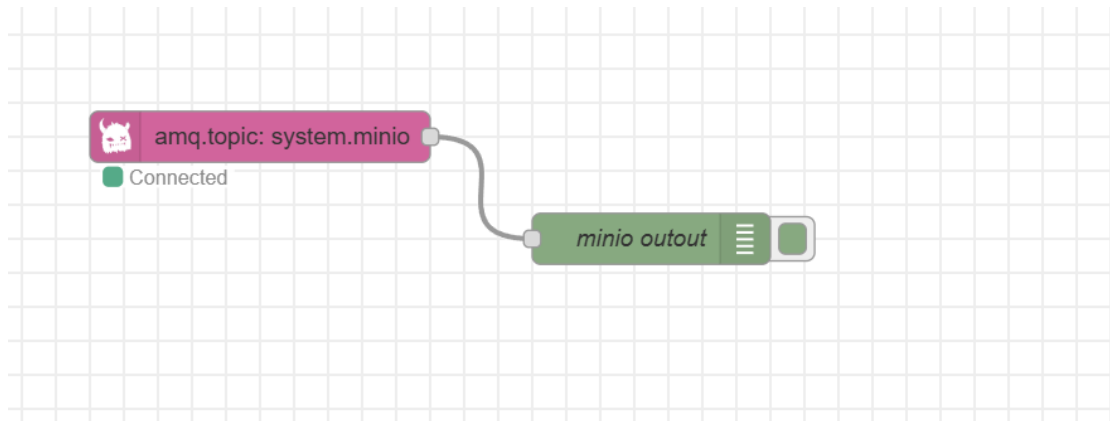


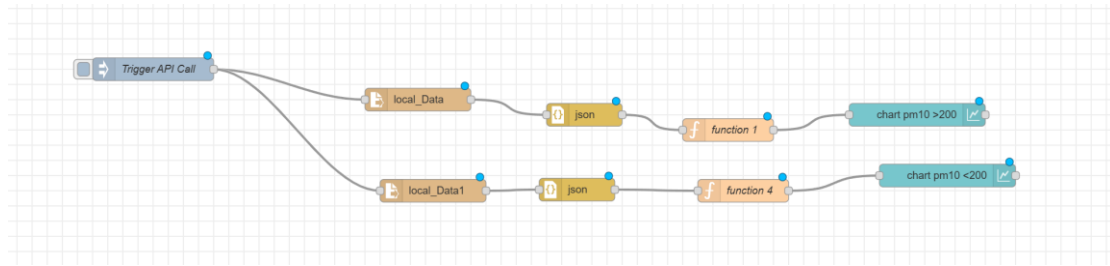
Figure 5 node-red and rabbitmq

Έχει γίνει διασύνδεση του Minio με το RabbitMQ, και κάθε φορά που προστίθεται μια πληροφορία στο bucket του Minio, λαμβάνουμε το παρακάτω μήνυμα στο Node-RED.

```
/2/2025, 5:29:22 μ.μ. node: minio outout
msg.payload : Object
▼object
  eventName: "s3:ObjectCreated:Put"
  key: "ergasia/local_Data"
  Records: array[1]
    0: object

/2/2025, 5:29:22 μ.μ. node: minio outout
msg.payload : Object
▼object
  eventName: "s3:ObjectCreated:Put"
  key: "ergasia/local_Data1"
  Records: array[1]
    0: object
      eventVersion: "2.0"
      eventSource: "minio:s3"
      awsRegion: ""
      eventTime: "2025-02-01T15:29:30.135Z"
      eventName: "s3:ObjectCreated:Put"
      userIdentity: object
        principalId: "minioadmin"
      requestParameters: object
        principalId: "minioadmin"
        region: ""
        sourceIPAddress: "127.0.0.1"
      responseElements: object
      s3: object
      source: object
```

Επομένη χρήση είναι η δημιουργία διαγραμμάτων.



Με το ξεκίνημα του trigger, διαβάζω τα δεδομένα που υπάρχουν στους φακέλους και, με τη χρήση της function, εξάγω τα πρώτα δεδομένα από κάθε φάκελο για να τα παρουσιάσω σε μια dashboard σελίδα.

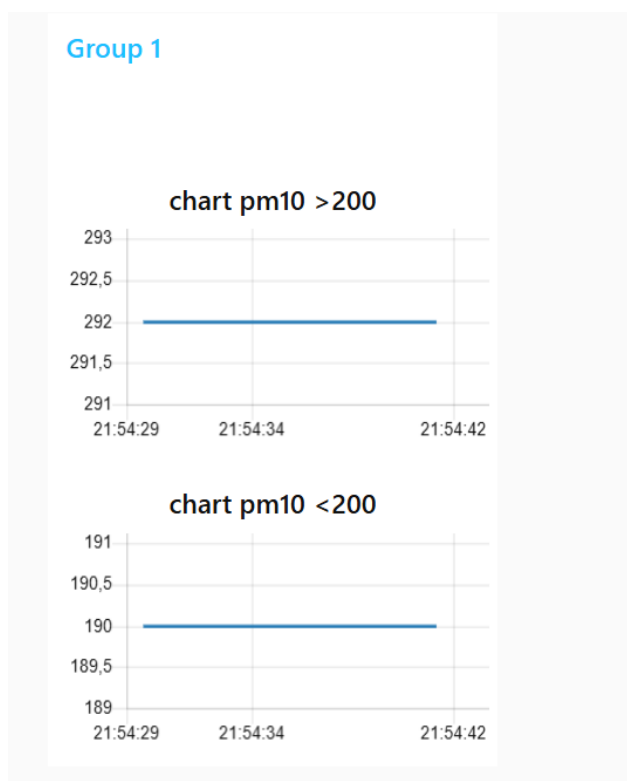


Figure 6 dashboard in node-red

Στο πάνω διάγραμμα φαίνεται τα πάνω από 200 pm10 και στο κάτω διάγραμμά τα κάτω από 200.

Το link για το dashboard είναι το [Node-RED Dashboard](#)

Επίσης έχει γίνει υλοποίηση του παρακάτω που είναι το RabbitMQ να μιλάει με το keycloak, και κάθε φορά που γίνεται ένα event στο keycloak να το λαμβάνω σαν message στο node-red

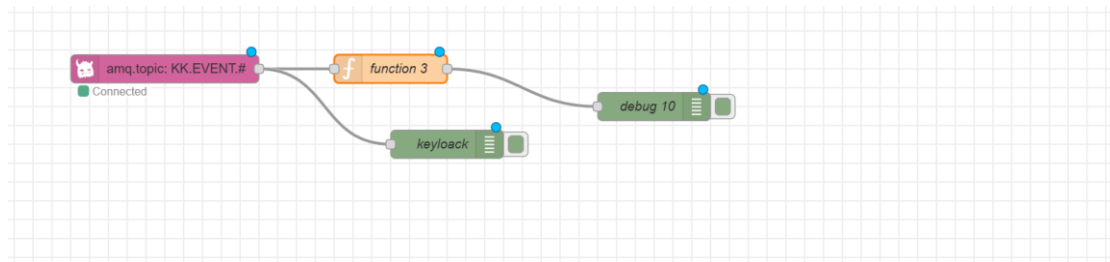


Figure 7 keycloak in node-red

```
1/2/2025, 7:37:02 π.μ. node: keycloak
msg.payload: Object
▼ object
  @class:
    "com.github.aznamier.keycloak.event.provider.EventClientNotificationMqMsg"
  time: 1738431432093
  type: "REFRESH_TOKEN"
  realmId: "a6710c78-0243-4430-b411-a43dede47262"
  clientId: "security-admin-console"
  userId: "51dcbd4d-9c18-4567-9a18-0a5b94c33e10"
  sessionId: "ca362a78-d7bf-4a3a-a732-e5e3de64f2d8"
  ipAddress: "172.19.0.1"
  ▼ details: object
    token_id: "6790e655-a364-4f66-811b-1598773b9908"
    grant_type: "refresh_token"
    refresh_token_type: "Refresh"
    access_token_expiration_time: "60"
    updated_refresh_token_id: "4e819a74-8599-46ea-b6d1-3474b07bfd3c"
    scope: "openid profile email"
    age_of_refresh_token: "162"
    refresh_token_id: "dc49725e-381d-4612-b296-478d31aded13"
    client_auth_method: "client-secret"
```

Figure 8 token from keycloak in node-red

Minio

Το πρόγραμμα Minio τρέχει σε Kubernetes. Έχω φτιάξει bucket ώστε να αποθηκεύονται τα δεδομένα που έρχονται από το node-red

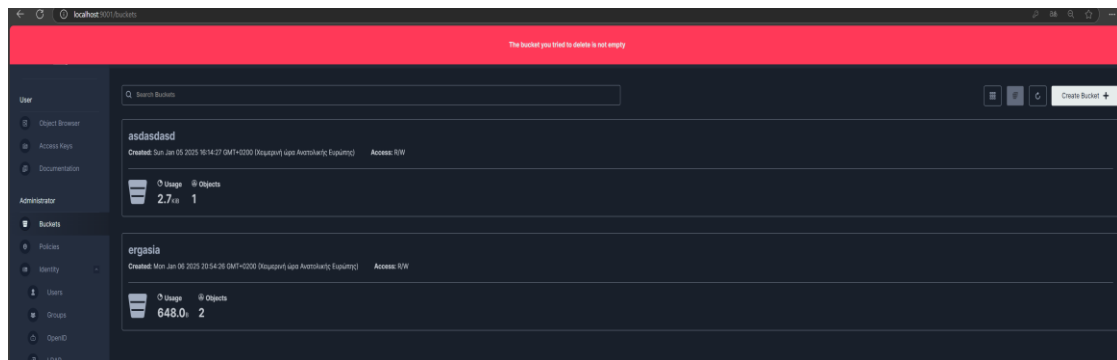


Figure 9 minio buckets

Τα παρακάτω αρχεία είναι αυτά που έχουν μπει από το node-red

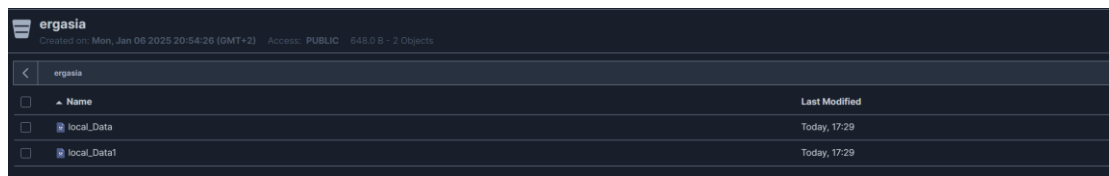


Figure 10 bucket data

Στο bucket ergasia έχω βάλει και events < put,get,delete> ώστε τα λαμβάνω events, μέσω του RabbitMQ.

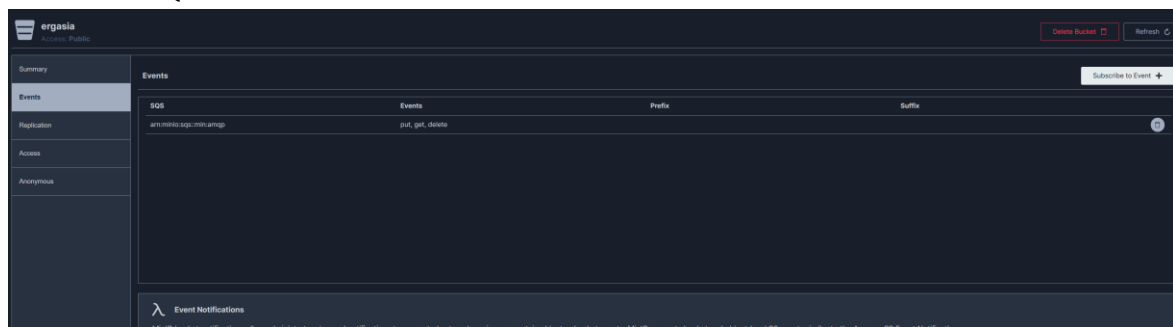


Figure 11 events in bucket

Το Minio επικοινωνεί με το Keycloak μέσω της παρακάτω διασύνδεσης, η οποία επιτρέπει την αυθεντικοποίηση κάθε χρήστη. Ο χρήστης διαθέτει μόνο δικαιώματα (read).

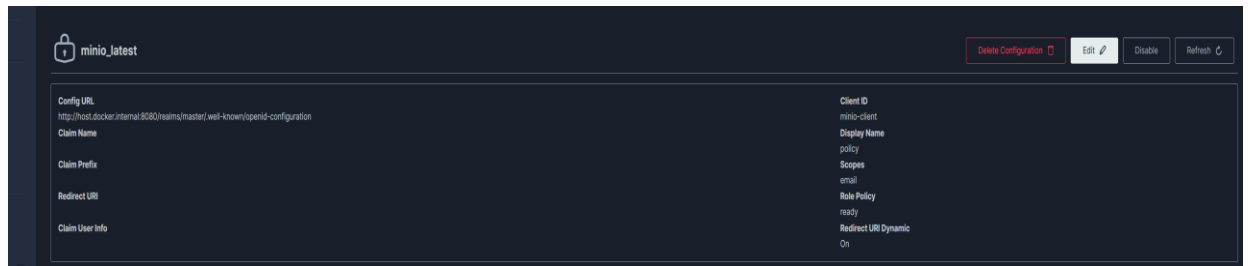


Figure 12 minio connection with keycloak

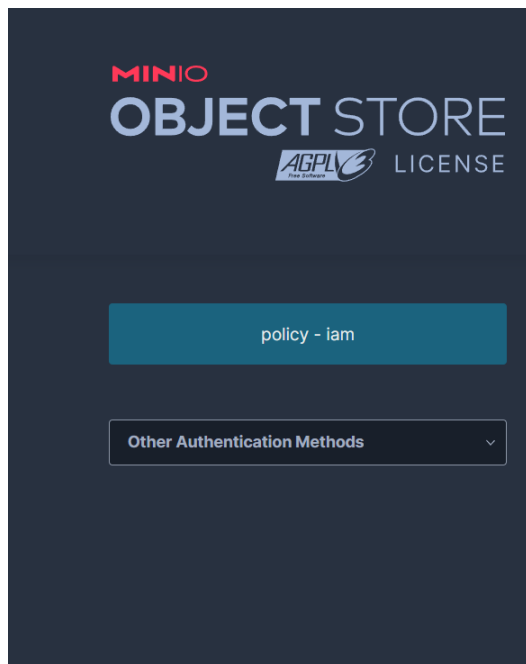


Figure 13 minio front page

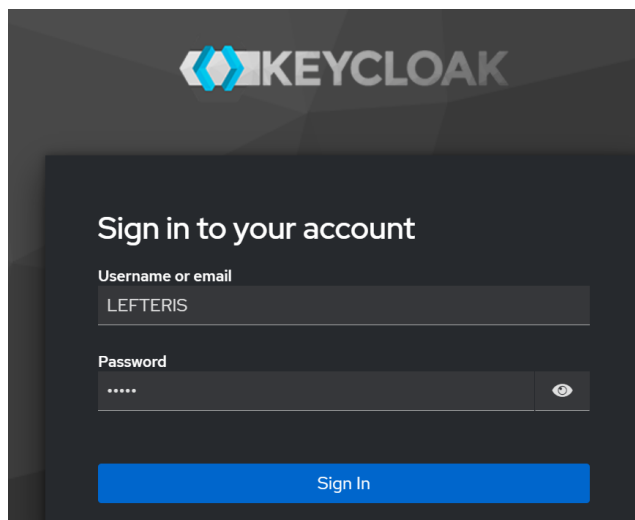


Figure 14 minio connection with keycloak

Το bucket που έχει πρόσβαση το minio μετά απο την αυθεντικοποίηση είναι με read δικαιώματα.

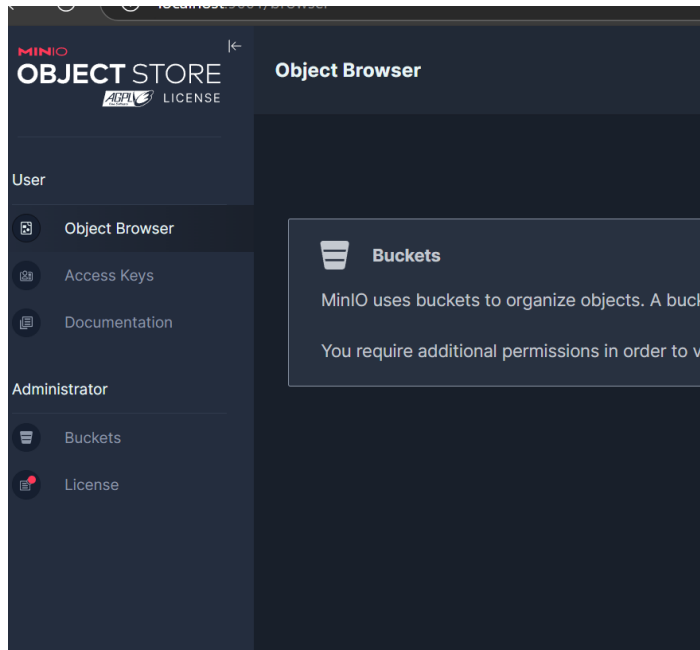


Figure 15 Minio login user with minimal rights

RABBITMQ

Το RabbitMQ τρέχει σε Docker. Ως χρήστης, έχω χρησιμοποιήσει τον προεπιλεγμένο (guest/guest), του οποίου τα credentials ορίζονται στο docker-compose file. Το πρώτο βήμα ήταν η δημιουργία του amq.topic, ο οποίος θα λαμβάνει δεδομένα από το Minio και το keycloak.

Exchanges

▼ All exchanges (9)

Pagination

Page 1 of 1 - Filter: ☐ Regex ?

Virtual host	Name	Type	Features	Message rate in	Message rate out	+/-
/	(AMQP default)	direct	D			
/	amq.direct	direct	D			
/	amq.fanout	fanout	D			
/	amq.headers	headers	D			
/	amq.match	headers	D			
/	amq.rabbitmq.trace	topic	D I			
/	amq.topic	topic	D	0.00/s	0.00/s	
/	keycloak.events	direct	D			
/	topic	topic	D			

Figure 16 Rabbitmq topic outside

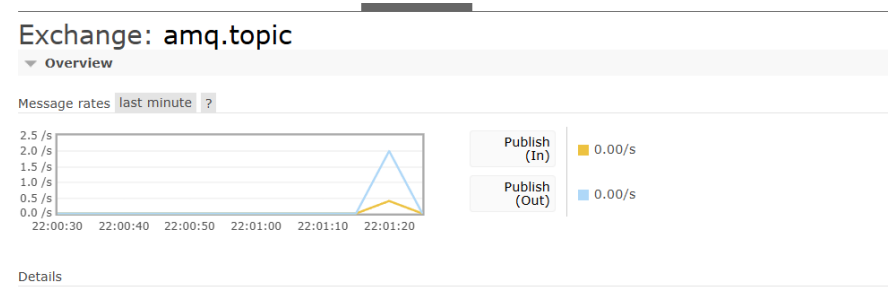


Figure 17 Rabbitmq topic inside

Μετά είναι η δημιουργία των 2 queue ώστε να τα κάνω bind την πληροφορία

Page 1 of 1 - Filter: ☐ Regex ?

Overview					Messages			Message rates		
Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
/	MINIO	classic	D Args	■ running	19	0	19	0.00/s		
/	amq.gen--Q6CIKwvp3b_fn0Ia3gheA	classic	D Args	■ running	19	0	19	0.00/s		
/	amq.gen-0RB4puE4G_iM_rcVSgd5fA	classic	D Args	■ running	19	0	19	0.00/s		
/	amq.gen-1MPj4l_A4Q9my6SfsG5ePA	classic	D Args	■ running	19	0	19	0.00/s		
/	amq.gen-RvgNYuAU-QpZpA3I3bOmvgg	classic	D Args	■ running	42	0	42	0.00/s		
/	amq.gen-TMHXfuqP3IYII3krW8k9jg	classic	D Args	■ running	42	0	42	0.00/s		
/	amq.gen-exLwaHIzmWc-iesDyZjssg	classic	D Args	■ running	0	0	0	0.00/s	0.00/s	0.00/s
/	amq.gen-v7L9xK_vaQzFN0uYdIg1QQ	classic	D Args	■ running	0	0	0	0.00/s	0.00/s	0.00/s
/	amq.gen-y1NwoIlo55PGefyMXEhljA	classic	D Args	■ running	42	0	42	0.00/s		
/	keyl	classic	D Args	■ running	42	0	42	0.00/s		
/	yh	classic	D Args	■ running	0	0	0			

Figure 18Rabbitmq Queues

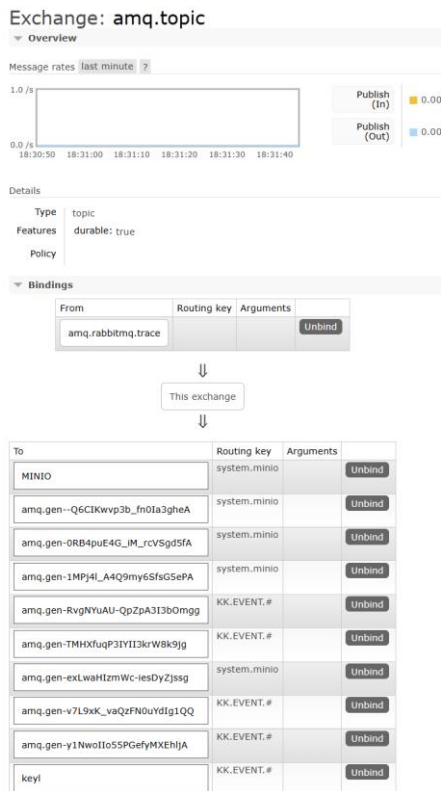


Figure 19 Rabbitmq inside topic with routing key

To system.minio είναι για το minio, και το kk.event# είναι για το keycloak

Keycloak.

Το keycloak είναι το πρόγραμμα που χρησιμοποιώ για γίνει το authentication ανάμεσα στο minio-keycloak, τρέχει σε docker. Για να φτιάξω την διασύνδεση ακολούθησα τις οδηγίες που υπάρχουν στο παρακάτω сайт. [Configure MinIO for Authentication using Keycloak — MinIO Object Storage for MacOS](#)

Πρώτο βήμα έφτιαξα ένα minio-client

The screenshot shows the Keycloak Admin Console interface for configuring a new client. The client is named 'minio-client' and is of type 'OpenID Connect'. The 'Settings' tab is selected, showing the 'General settings' section. The 'Client ID' is 'minio-client'. The 'Name' and 'Description' fields are empty. The 'Always display in UI' toggle is turned 'On'. The 'Access settings' section shows the 'Root URL' and 'Home URL' both set to 'http://localhost:9001'. The 'Valid redirect URIs' field contains an asterisk (*), with a link to 'Add valid redirect URIs'. The 'Valid post logout redirect URIs' field is empty, with a link to 'Add valid post logout redirect URIs'. The 'Web origins' field is empty, with a link to 'Add web origins'. The 'Admin URL' is set to 'http://localhost:9001'.

minio-client OpenID Connect

Clients are applications and services that can request authentication of a user.

Settings Keys Credentials Roles Client scopes Sessions Advanced

General settings

Client ID * ⓘ minio-client

Name ⓘ

Description ⓘ

Always display in UI ⓘ ☒ On

Access settings

Root URL ⓘ http://localhost:9001

Home URL ⓘ http://localhost:9001

Valid redirect URIs ⓘ *

+ Add valid redirect URIs

Valid post logout redirect URIs ⓘ

+ Add valid post logout redirect URIs

Web origins ⓘ

+ Add web origins

Admin URL ⓘ http://localhost:9001

Figure 20 keycloak minio-client

Επόμενο βήμα ήταν να φτιάξω ένα client scope μαζί με τα κατάλληλα mapper στο policy

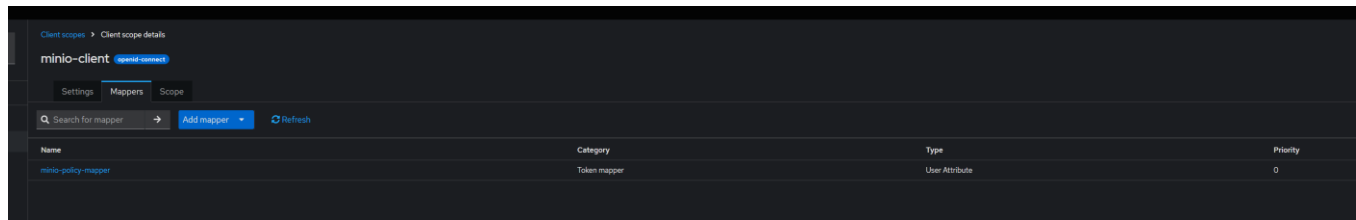


Figure 21 keycloak client scope

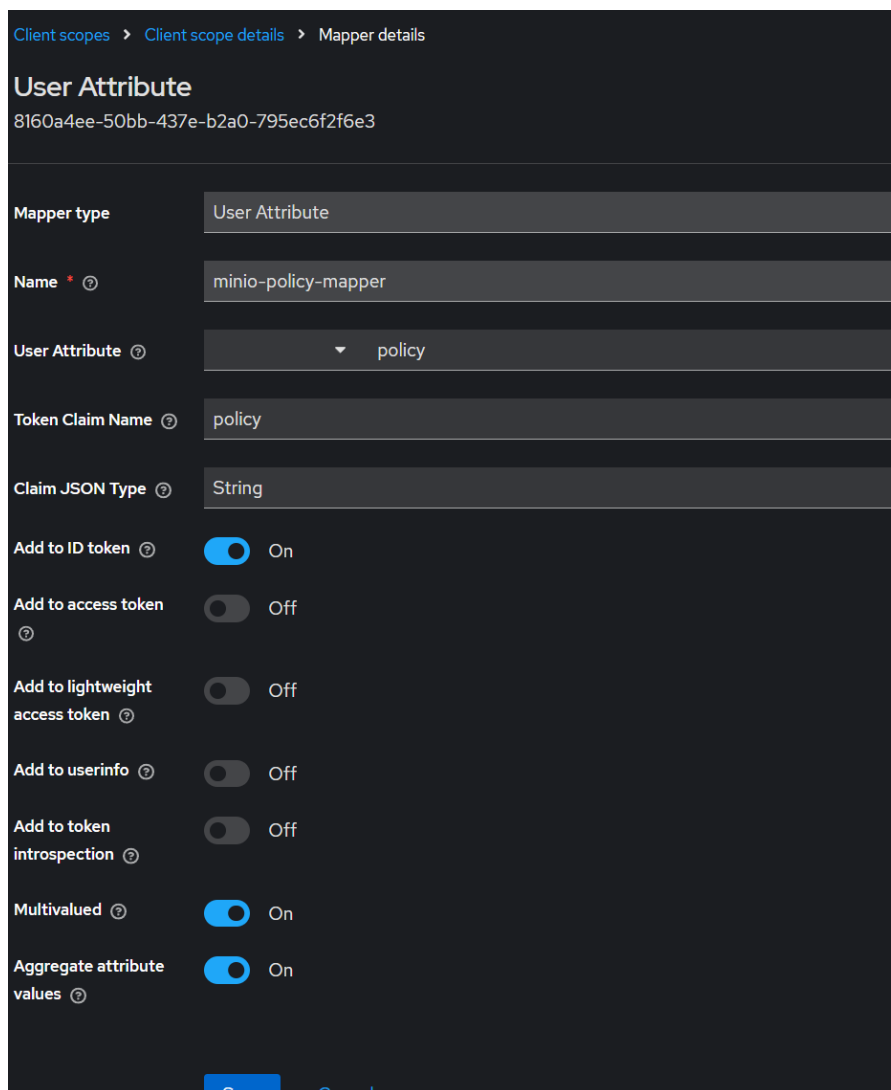


Figure 22 Keycloak mapper details

Ο χρήστη με policy read-only , για να γίνεται σύνδεση με το Minio.

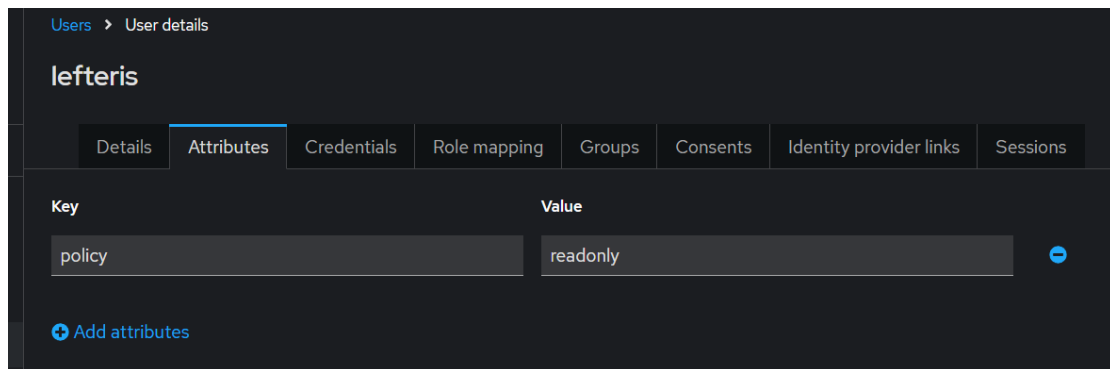


Figure 23 keycloak User policy

Για την διασύνδεση του keycloak με το RabbitMQ χρησιμοποίησα το [GitHub - aznamier/keycloak-event-listener-rabbitmq](https://github.com/aznamier/keycloak-event-listener-rabbitmq)

Στην αρχή έβαλα το κατάλληλο jar αρχείο μέσα στο σύστημα του keycloak στο path /opt/keycloak/providers/keycloak-to-rabbit-3.0.5.jar

Επόμενο βήμα ήταν να το προσθέσω στο realm σαν event

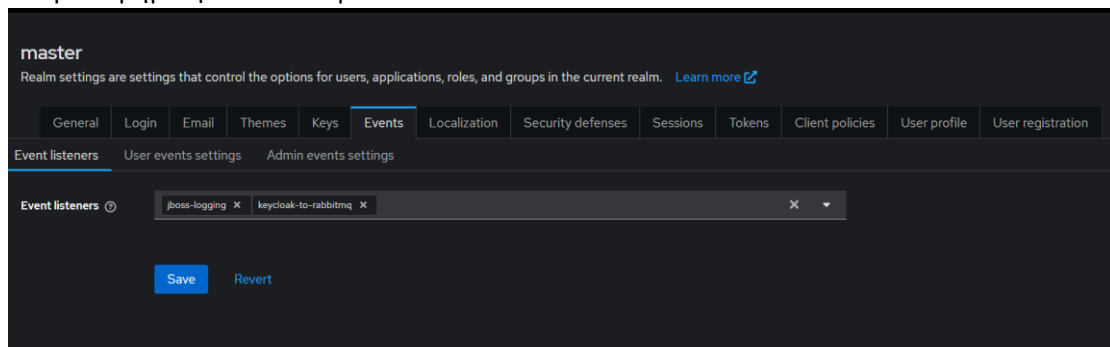


Figure 24 Keycloak event add

Επιπλέον έπρεπε να βάλω και τα παρακάτω στο docker_compose file, ώστε να γίνει η διασύνδεση


```

keycloak:
  image: quay.io/keycloak/keycloak:latest
  container_name: keycloak
  environment:
    - KC_BOOTSTRAP_ADMIN_USERNAME=admin
    - KC_BOOTSTRAP_ADMIN_PASSWORD=admin
    # RabbitMQ connection settings
    - KK_TO_RMQ_URL=rabbitmq
    - KK_TO_RMQ_PORT=5672
    - KK_TO_RMQ_VHOST=/
    - KK_TO_RMQ_USERNAME=guest
    - KK_TO_RMQ_PASSWORD=guest
    - KK_TO_RMQ_USE_TLS=false
    - KK_TO_RMQ_EXCHANGE =amq.topic
    - KK_TO_RMQ_KEY_STORE =
    - KK_TO_RMQ_KEY_STORE_PAS=
    - KK_TO_RMQ_TRUST_STORE=
    - KK_TO_RMQ_TRUST_STORE_PASS=
  ports:
    - "8080:8080"

```

Figure 25 keycloak docker file for RabbitMq

Docker and Kubernetes

Αρχικά, εγκατέστησα το Docker στον υπολογιστή μου μαζί με την επιλογή του Kubernetes.

Το επόμενο βήμα είναι η δημιουργία του docker-compose file, στο οποίο έχω χρησιμοποιήσει τα προεπιλεγμένα (default) images.

Στο image του node-red έχω προσθέσει την επιλογή - ./nodered-data:/data ώστε να μπορώ να αποθηκεύω τα flows κάθε φορά

Η επιλογή mynetwork είναι σε όλα τα images διότι είναι ο τρόπος να μιλάνε μεταξύ τους.

```
services:
  node-red:
    image: nodered/node-red:lates
    container_name: node-red
    ports:
      - "1880:1880"
    volumes:
      - ./nodered-data:/data
    networks:
      - mynetwork
```

Figure 26 node-red in docker compuse file

στο RabbitMQ έχω περάσει την επιλογή για να έχει έτοιμο ένα χρήστη, τον guest και πάλι την επιλογή για το network.

```
rabbitmq:
  image: rabbitmq:management
  container_name: rabbitmq
  ports:
    - "5672:5672"
    - "15672:15672"
  environment:
    - RABBITMQ_DEFAULT_USER=guest
    - RABBITMQ_DEFAULT_PASS=guest
  networks:
    - mynetwork
```

Figure 27 rabbitmq in docker compose file

Το Keycloak παρέχει τη δυνατότητα δημιουργίας χρήστη για τη σύνδεση στην εφαρμογή. Επίσης, διαθέτει ρυθμίσεις δικτύου ώστε να επιτρέπεται η επικοινωνία μεταξύ των υπηρεσιών. Τέλος, έχουν προστεθεί τα απαραίτητα environment variables για τη σύνδεσή του με το RabbitMQ.

```
keycloak:
  image: quay.io/keycloak/keycloak:latest
  container_name: keycloak
  environment:
    - KC_BOOTSTRAP_ADMIN_USERNAME=admin
    - KC_BOOTSTRAP_ADMIN_PASSWORD=admin
    # RabbitMQ connection settings
    - KK_TO_RM_Q_URL=rabbitmq
    - KK_TO_RM_Q_PORT=5672
    - KK_TO_RM_Q_VHOST=/
    - KK_TO_RM_Q_USERNAME=guest
    - KK_TO_RM_Q_PASSWORD=guest
    - KK_TO_RM_Q_USE_TLS=false
    - KK_TO_RM_Q_EXCHANGE =amq.topic
    - KK_TO_RM_Q_KEY_STORE =
    - KK_TO_RM_Q_KEY_STORE_PAS=
    - KK_TO_RM_Q_TRUST_STORE=
    - KK_TO_RM_Q_TRUST_STORE_PASS=
  ports:
    - "8080:8080"
  volumes:
    - ./keycloak/providers:/opt/keycloak/providers
  command:
    - start-dev
  depends_on:
    - rabbitmq
  networks:
    - mynetwork

networks:
  mynetwork:
    driver: bridge
```

Figure 28 keycloak in docker compose file

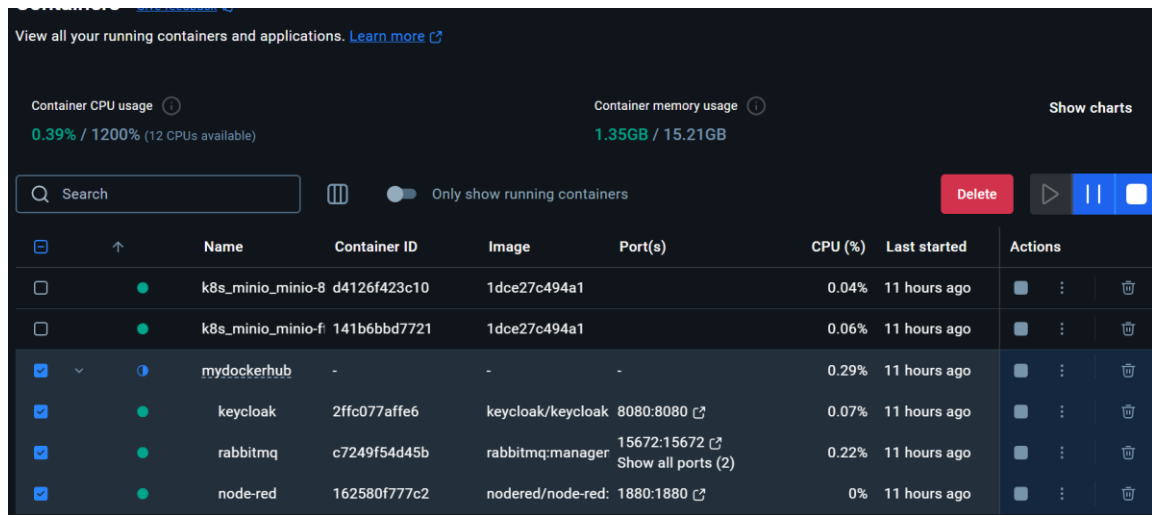


Figure 29 docker in my pc with nodes and kubernetes

Kubernetes

Για την δημιουργία του Kubernetes, χρησιμοποίησα το Kompose που κάνει convert to docker compose file, σε kubernetes files, και επίσης φτιάχνει και το service.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: minio-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    io.kompose.service: minio
  name: minio
spec:
  replicas: 1
  selector:
    matchLabels:
      io.kompose.service: minio
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        io.kompose.service: minio
    spec:
      containers:
        - args:
            - server
            - --console-address
            - :9001
            - /data
          env:
            - name: MINIO_ROOT_PASSWORD
              value: minioadmin
            - name: MINIO_ROOT_USER
              value: minioadmin
            - name: MINIO_BROWSER_LOGIN_ANIMATION
              value: "off"
          image: minio/minio:latest
          name: minio
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 9000
              protocol: TCP
            - containerPort: 9001
              protocol: TCP
          volumeMounts:
            - name: minio-storage
              mountPath: /data
      volumes:
        - name: minio-storage
          persistentVolumeClaim:
            claimName: minio-pvc
      restartPolicy: Always
```

Figure 30 kubernetes file

Service

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kompose.cmd: C:\Users\Istev\AppData\Local\Microsoft\WinGet\Packages\Kubernetes.kompose_Microsoft.Winget.Source_8wekyb3d8bbwe\kompose.exe convert
    kompose.version: 1.35.0 (9532ceef3)
  labels:
    io.kompose.service: minio
  name: minio
spec:
  ports:
    - name: "9000"
      port: 9000
      targetPort: 9000
    - name: "9001"
      port: 9001
      targetPort: 9001
  selector:
    io.kompose.service: minio
```

Figure 31 kubernetes minio service file

Επίσης παρατήρησα ότι κάθε φορά που έκανα restart το cluster με replica=0, χανόντουσαν όλες οι ρυθμίσεις στο Minio, όποτε έφτιαξα ένα minio-pv-pvc.yaml ώστε να κάνει mount ένα storage στον υπολογιστή μου

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: minio-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /c/Users/user/Documents/my-docker-project/mydockerhub
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: minio-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
=====
apiVersion: v1
kind: PersistentVolume
metadata:
  name: minio-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /c/Users/user/Documents/my-docker-project/mydockerhub
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: minio-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
>>>>>> 893f0dfad5b9a897a23562186ec4612b08f623c5
```

Figure 32 Minio Storage file in kubernetes

Για να έχω πρόσβαση στην εφαρμογή χρησιμοποιώ

- port-forward svc/minio 9001:9001
- port-forward svc/minio 9000:9000

```
lsteve@user MINGW64 ~  
$ kubectl get pods  
NAME                READY   STATUS    RESTARTS   AGE  
minio-ffdbb899c-nhpsx 1/1     Running   4 (24d ago) 26d  
  
lsteve@user MINGW64 ~  
$ kubectl get svc  
NAME                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)              AGE  
kubernetes          ClusterIP   10.96.0.1    <none>        443/TCP              28d  
minio                ClusterIP   10.96.139.1  <none>        9000/TCP,9001/TCP    26d  
  
lsteve@user MINGW64 ~  
$
```

Figure 33 kubectl in my pc

<https://github.com/secretman12/mydockerhub/tree/main> to link με το github