

Projekat 3 – Uputstvo

Senzor service :

Senzor service je pisan u Node.js-u i služi za slanje podataka. Ovaj mikroservis prvo vrši čitanje podataka iz dataseta. Nakon toga se preko EMQX brokera podaci šalju preko topica na svakih 5 sekundi.

Filter / Analytics service :

Filter service je zadužen za primanje podataka koji su poslata preko senzor service i određenog topica. Za pristigle podatke se vrši izračunavanje srednjih vrednosti. Nakon izračunavanja se srednje vrednosti koje su dobijene se šalju preko brokera na novi topic. Service je pisan u .Net-u 7.

Dashboard service :

Dashboard service je pisan u Node.js-u i služi za upis podataka koji dolaze sa filter servisa u time series bazu podataka odnosno u InfluxDB. Podaci koji pristižu preko brokera se upisuju u Influx bazu koja zajedno sa Grafanom vrši vizuelizaciju tih podataka.

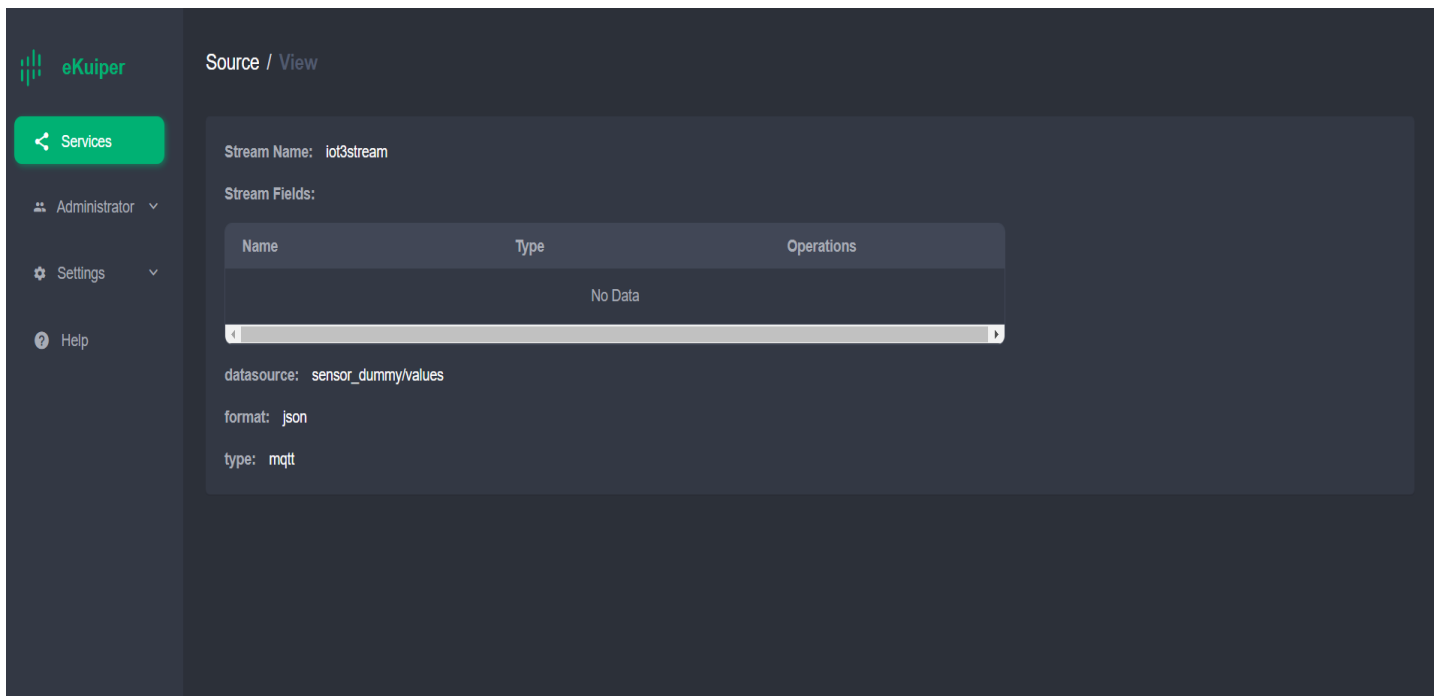
EKuiper service :

U docker compose fajlu je kao default adresa za eKuiper postavljena lokalna adresa mog racunara zbog uspostavljanja konekcija svih servisa na istu mrežu, a i javlja se problem prilikom povezivanja na localhost. Takodje eKuiper dashboard je moguće otvoriti na sledecoj adresi: <http://192.168.99.100:9082>. Bitno je napomenuti da je potrebno da svi ostali kontejneri budu pokrenuti pre nego sto se pokrene sam eKuiper service. Logovanje se vrši sledecim podacima :

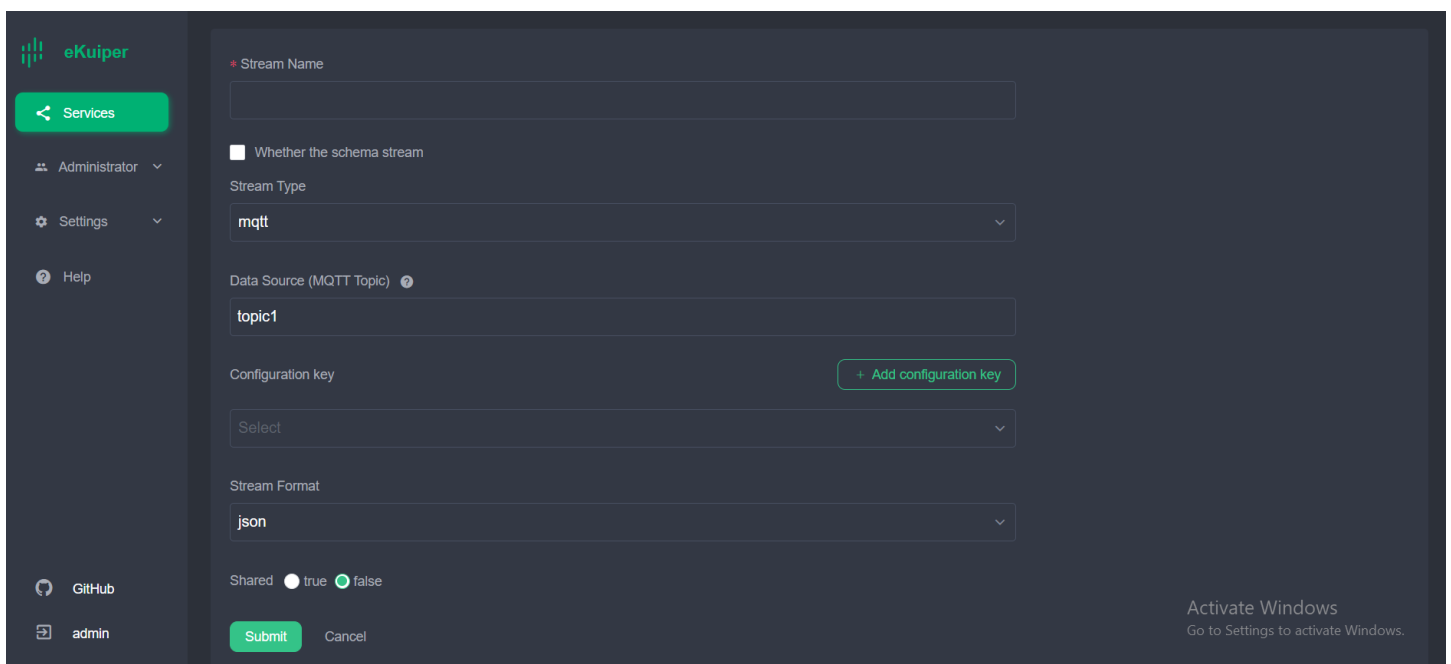
Username : admin

Password: public

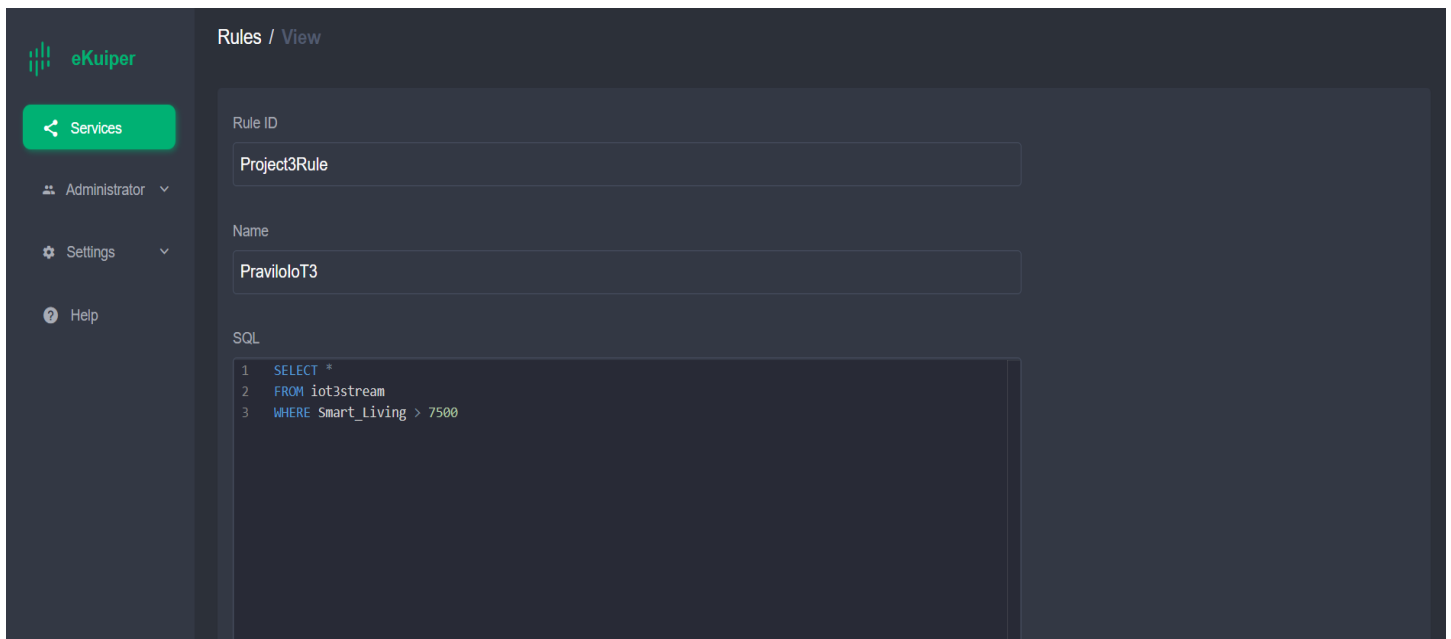
Nakon logovanja na eKuiper dashboard potrebno je kreirati data stream koji će primati podatke sa naseg mikroservisa . Potrebno je uneti sledece podatke vezane za data stream:



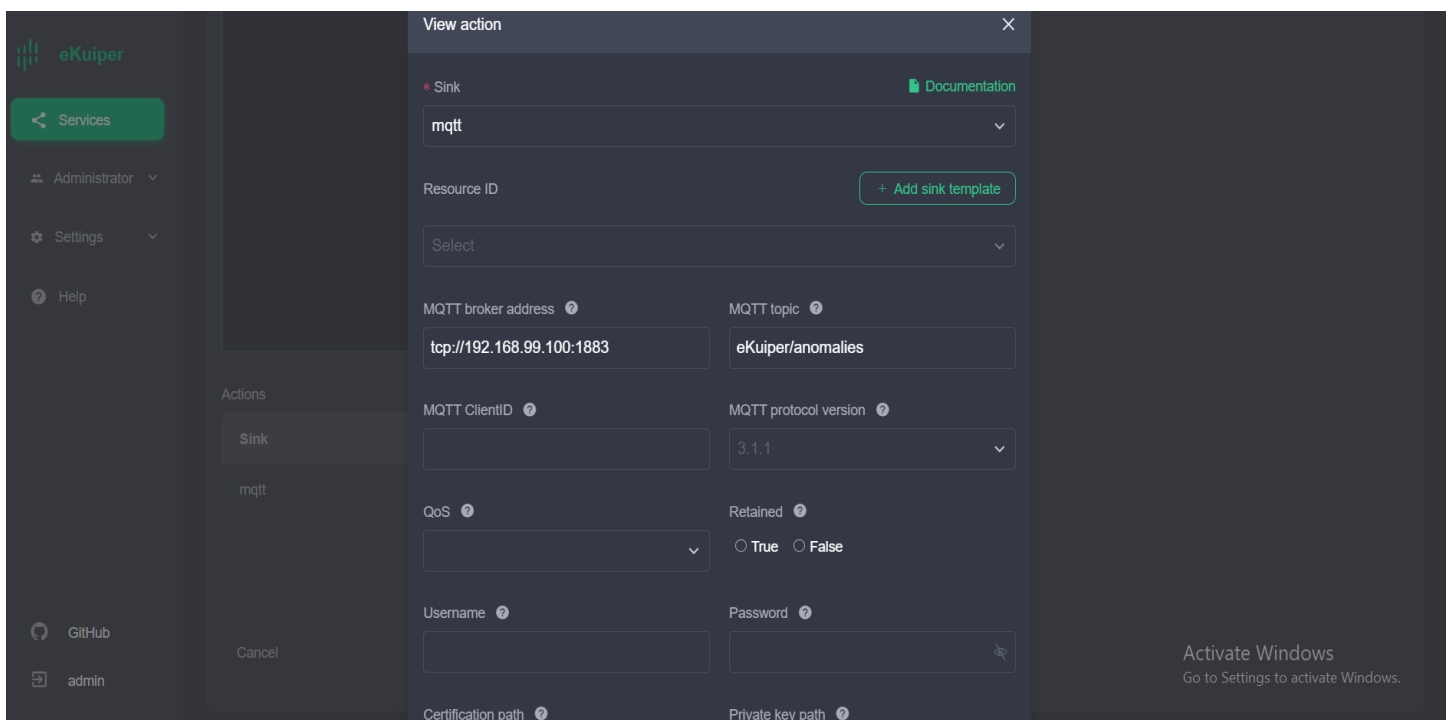
Forma koja se otvara nakon klika na dugme za kreiranje servisa je sledeca :



Nakon kreiranja streama potrebno je kreirati pravilo po kome ce se vrsiti filtriranje podataka :



Osim ovog pravila za filtriranje potrebno je dodati i akciju klikom na dugme +Add sa sledecim podacima :



Nakon ovoga potrebno je da pravilo bude aktivno kako bi eKuiper service mogao da se koristi.

Command service :

Command service je takodje pisan u Node.js-u i koristi express. Ovaj mikroservis služi za prijem poruka koje dobija filtriranjem preko eKuiper service i preko web socketa vrši njihovo slanje na klijentskoj web aplikaciji. Šalju se podaci od interesa koji se dobijaju eKuiper filterom, i vrši njihovo dodatno filtriranje preko indikatora u samom servisu gde se posmatra parametar podataka koji se odnosi na smart_mobility i imamo tri vrste indikacije, crvena boja je

za parametre koji nisu zadovoljavajući, žuta je za parametra koji su u nekoj meri zadovoljavajući dok je zelena boja indikatora za parametre koji su zadovoljavajući.

Izgled podataka koji pristižu na klijentsku web aplikaciju :

Event Display										
City	Country	SmartCity Index	SmartCity Index rel	Smart Economy	Smart Environment	Smart Government	Smart Living	Smart Mobility	Smart People	Indicator
Vantaa	Finland	6542	70	7760	6444	5722	8710	5223	5500	
Perth	Australia	5885	-587	3370	4026	5134	9620	5275	4810	
Jyväskylä	Finland	6448	-24	8175	6624	5852	8710	3900	6388	
Tokyo	Japan	5914	-558	1800	4516	5476	9320	5275	4158	
Melbourne	Australia	6061	-412	2780	3906	5848	9620	5275	5188	
Adelaide	Australia	5892	-581	3410	4148	5792	9620	5275	4678	
Tampere	Finland	6736	263	8365	6534	6696	8710	5203	5293	
Hämeenlinna	Finland	6503	31	8400	6746	6512	8710	4567	4590	
Oulu	Finland	6816	344	8135	6488	5642	8710	5258	7105	
Sydney	Australia	6442	-30	5075	3890	5932	9620	5275	7305	
Joensuu	Finland	6703	230	8325	6730	6356	8710	5045	5463	
Espoo	Finland	6804	332	7950	6458	6256	8710	5342	6503	
Turku	Finland	6676	204	8100	6504	5986	8710	4530	7050	
Geneva	Switzerland	6431	-41	6150	8706	5326	7960	4870	5650	

InfluxDB i Grafana

Kada sa pokrene docker-compose fajl pokrecu se kontejneri koji se odnose na InfluxDB i Grafanu. Potrebno je ući na sledeću adresu kako bi se pristupilo dashboardu InfluxDB-a : <http://192.168.99.100:8086> , nakon toga se otvara stranica sa prijavljivanje gde se unose podaci kao što su username, password, organisation i bucket , potrebno je uneti sledece podatke :

Username : Stefan

Password : adminadmin

Organization : organization

Bucket : iot3

Izgled forme u kojoj se unose ovi podaci :

Welcome — Initial User Setup — Complete

Setup Initial User

You will be able to create additional Users, Buckets and Organizations later

Username

Stefan

Password

Confirm Password

Initial Organization Name ?

organization

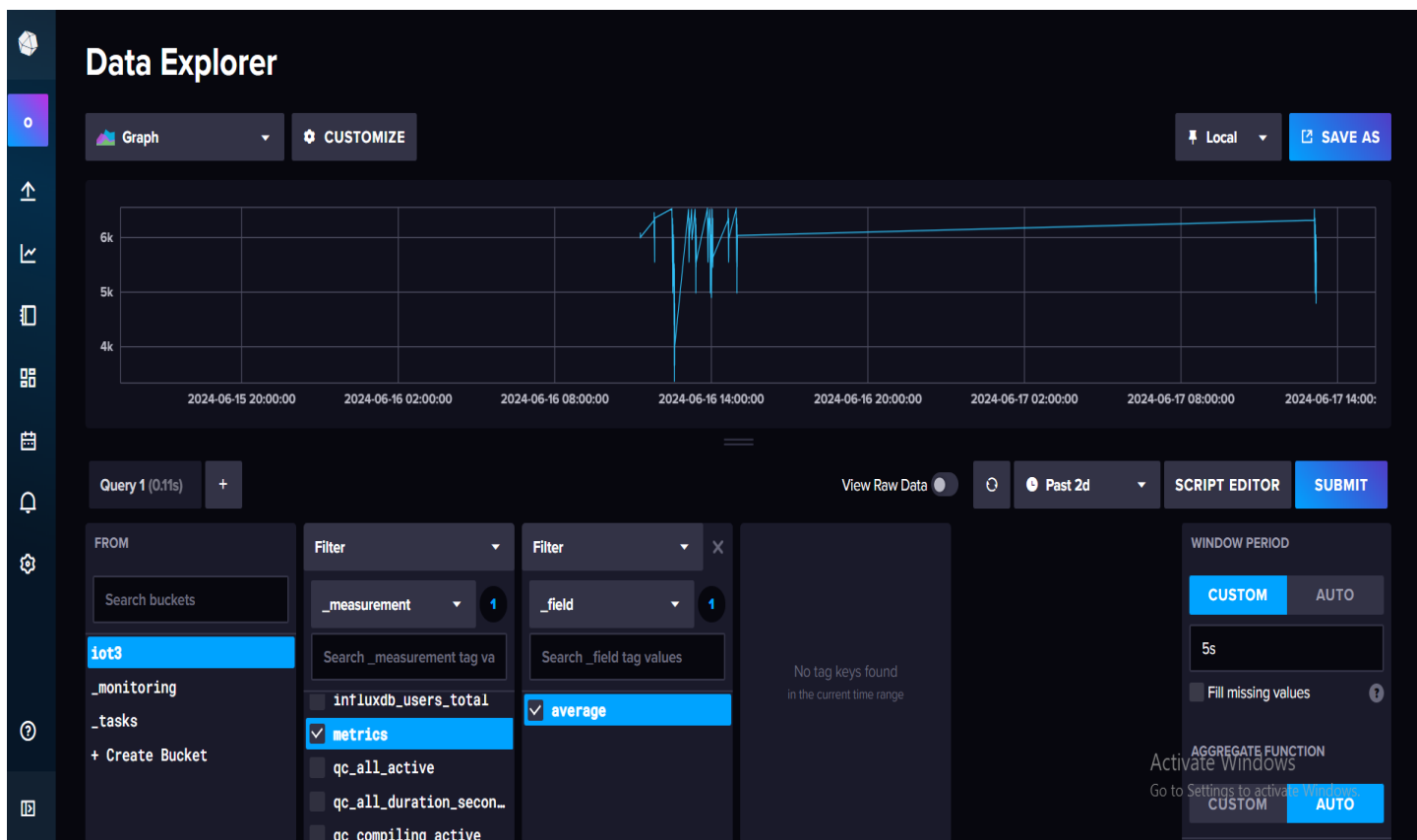
Initial Bucket Name ?

iot3

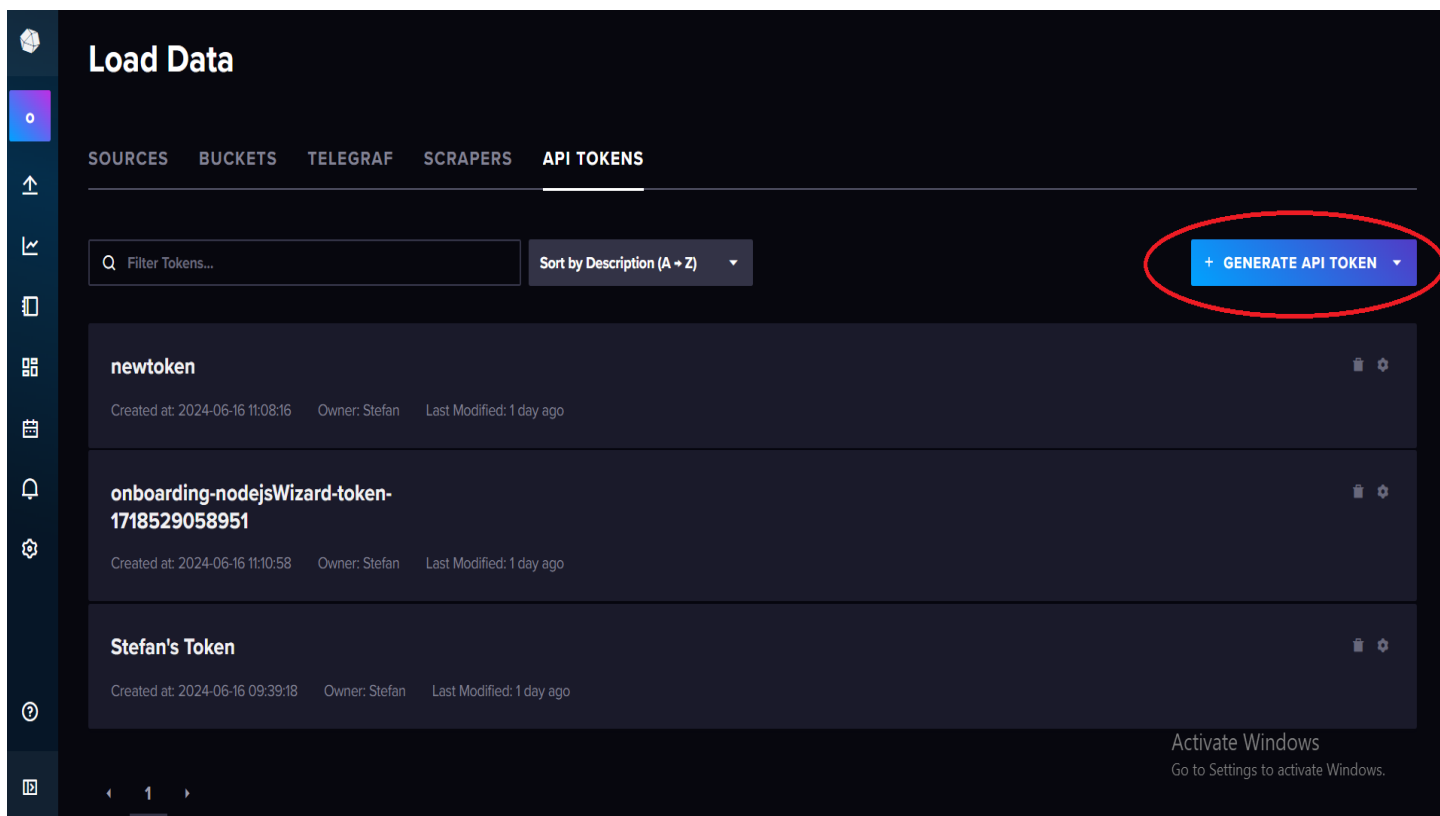
Activate Windows
Go to Settings to activate Windows.

CONTINUE

Nakon dodavanja podataka pomoću našeg mikroservisa kada uđemo u deo Data Explorer možemo videti vizuelizaciju naših podataka koji su upisani u izvesnom vremenskom periodu:



Takođe potrebno je kreirati token pristupa koji se koristi u našem mikroservisu prilikom povezivanja kao i Grafani kako bi se povezali sa InfluxDB bazom. Kreiranje api tokena se vrši na sledeći način :



Klikom na dugme označeno dugme se vrši generisanje tokena , pritom token je vidljiv samo jednom tako da je bolje upamtiti ga negde.

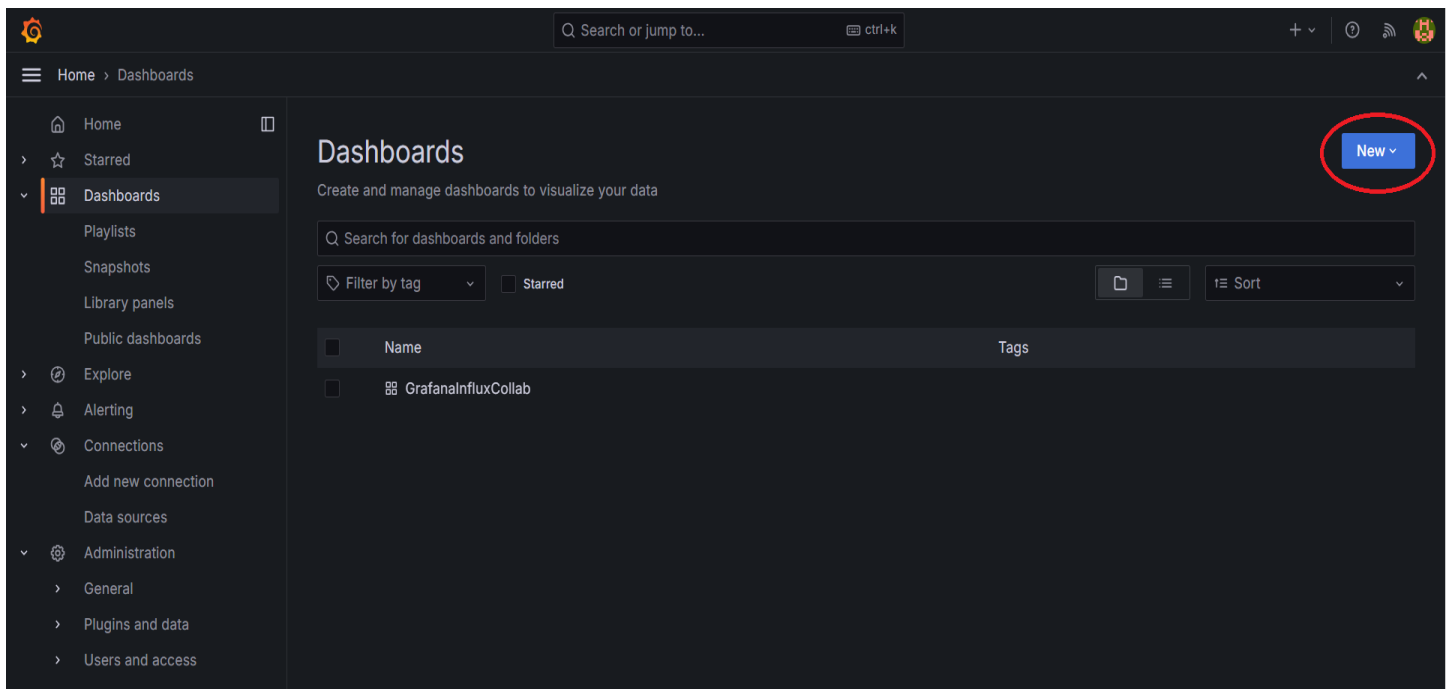
Što se tiče Grafane, njenom dashbordu se pristupa na sledećoj adresi :

<http://192.168.99.100:3000> , logujemo se kao :

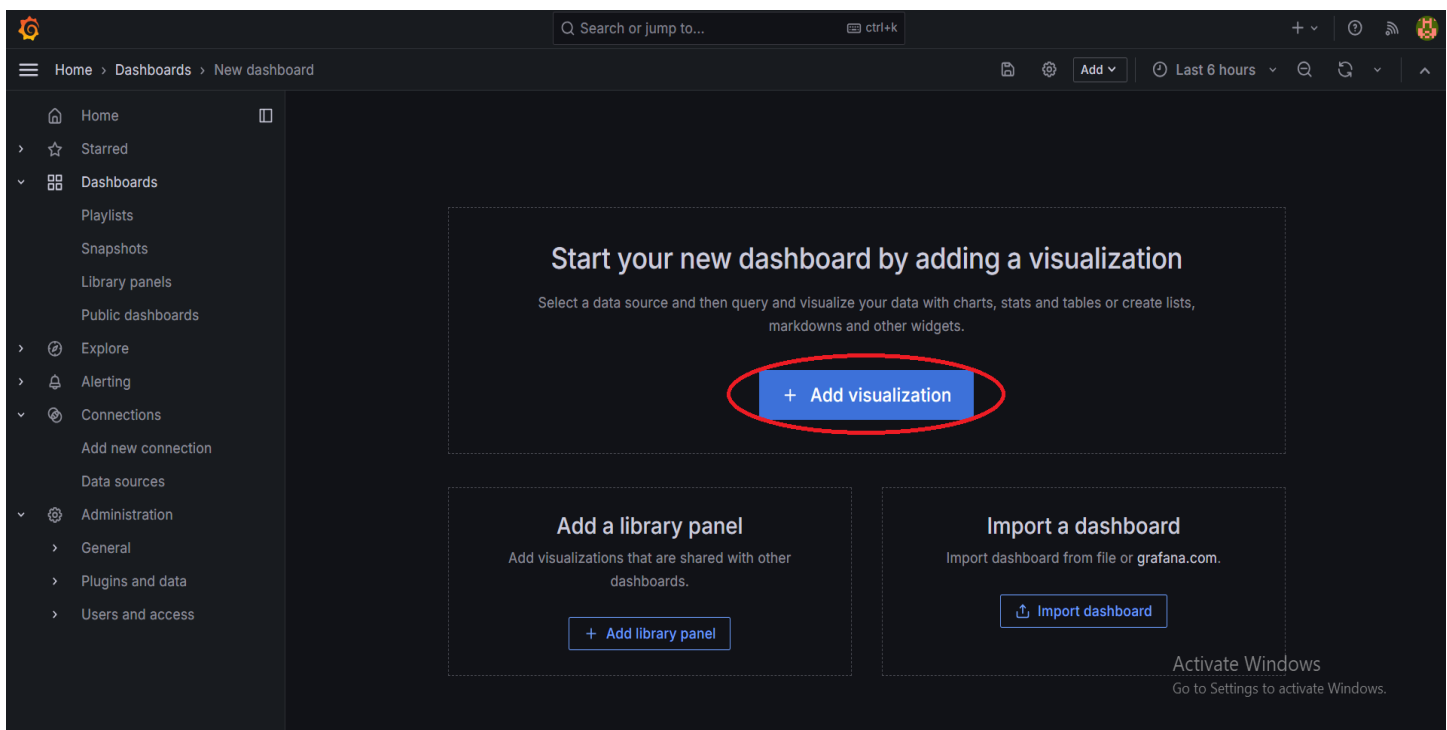
Username : admin

Password : admin

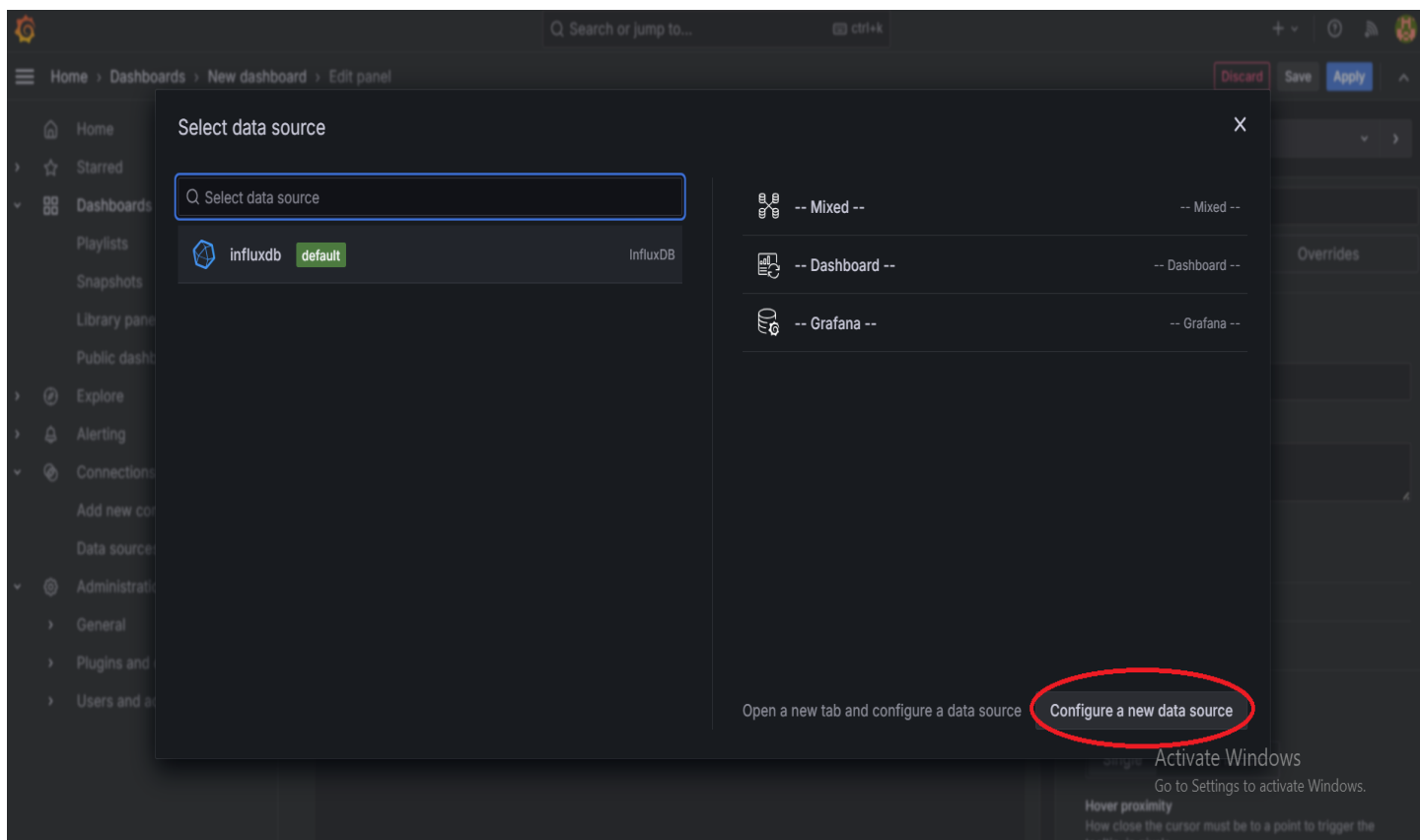
Šifru je moguće promeniti novom , ali ovaj korak može da se preskoči klikom na dugme skip koje se nalazi ispod polja za šifru. Nakon što smo se prijavili potrebno je otići u deo koji se odnosi na dashboard gde je potrebno kreirati novi dashboard na sledeći način :



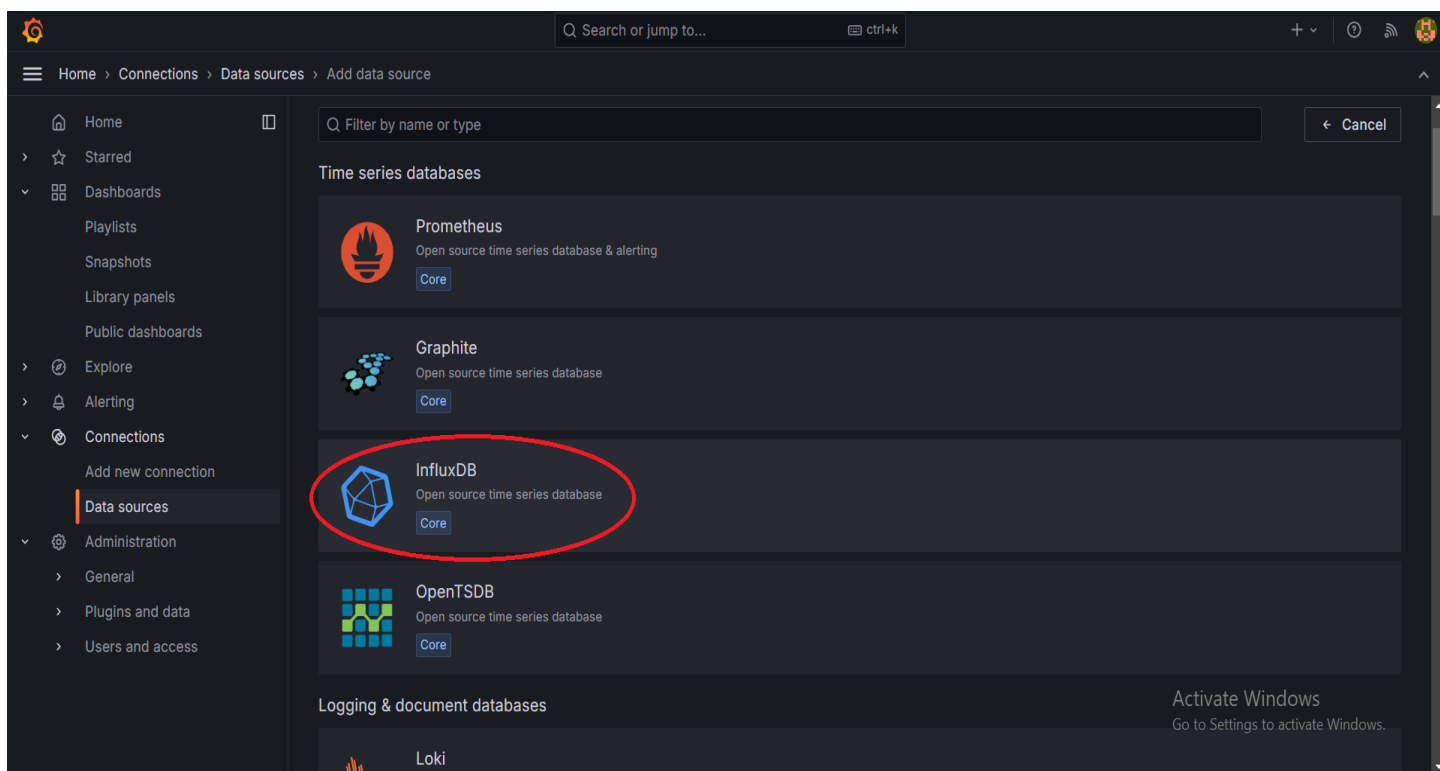
Kada kliknemo na dugme new biramo opciju New dashboard. Nakon toga se otvara sledeća forma :



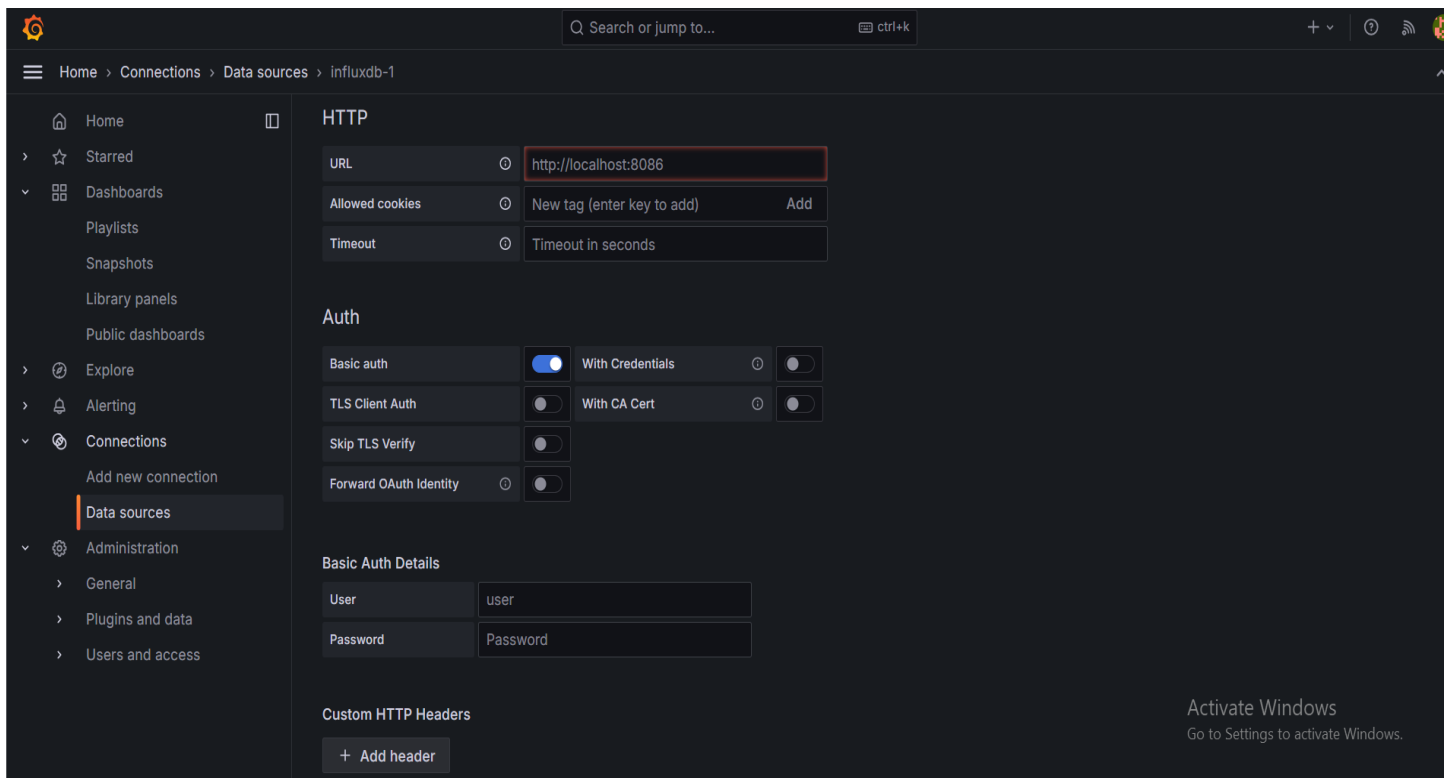
Nakon toga je potrebno dodati novu vizuelizaciju , i otvara se sledeća forma :



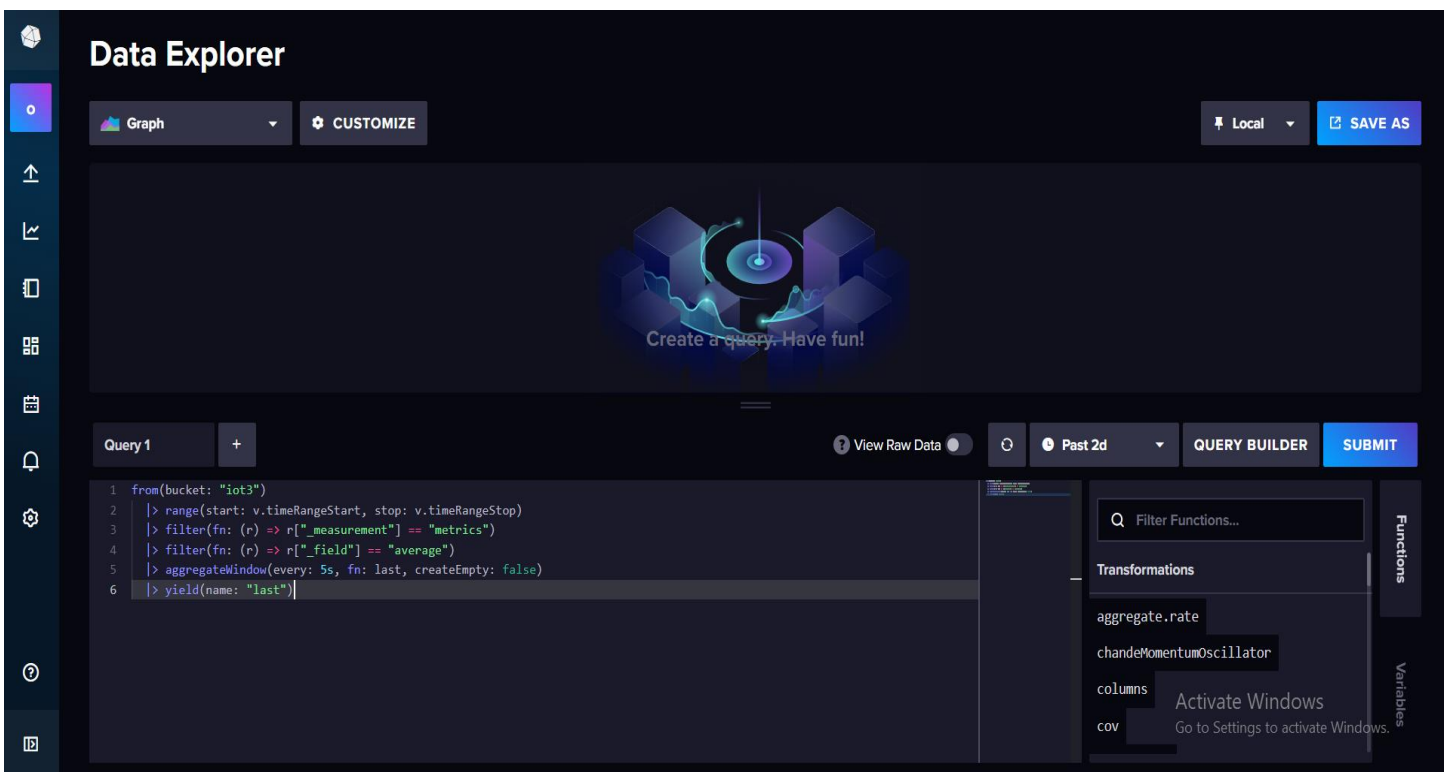
Ovde je potrebno ići na konfiguraciju novog dashboarda, nakon čega se otvara nova forma :



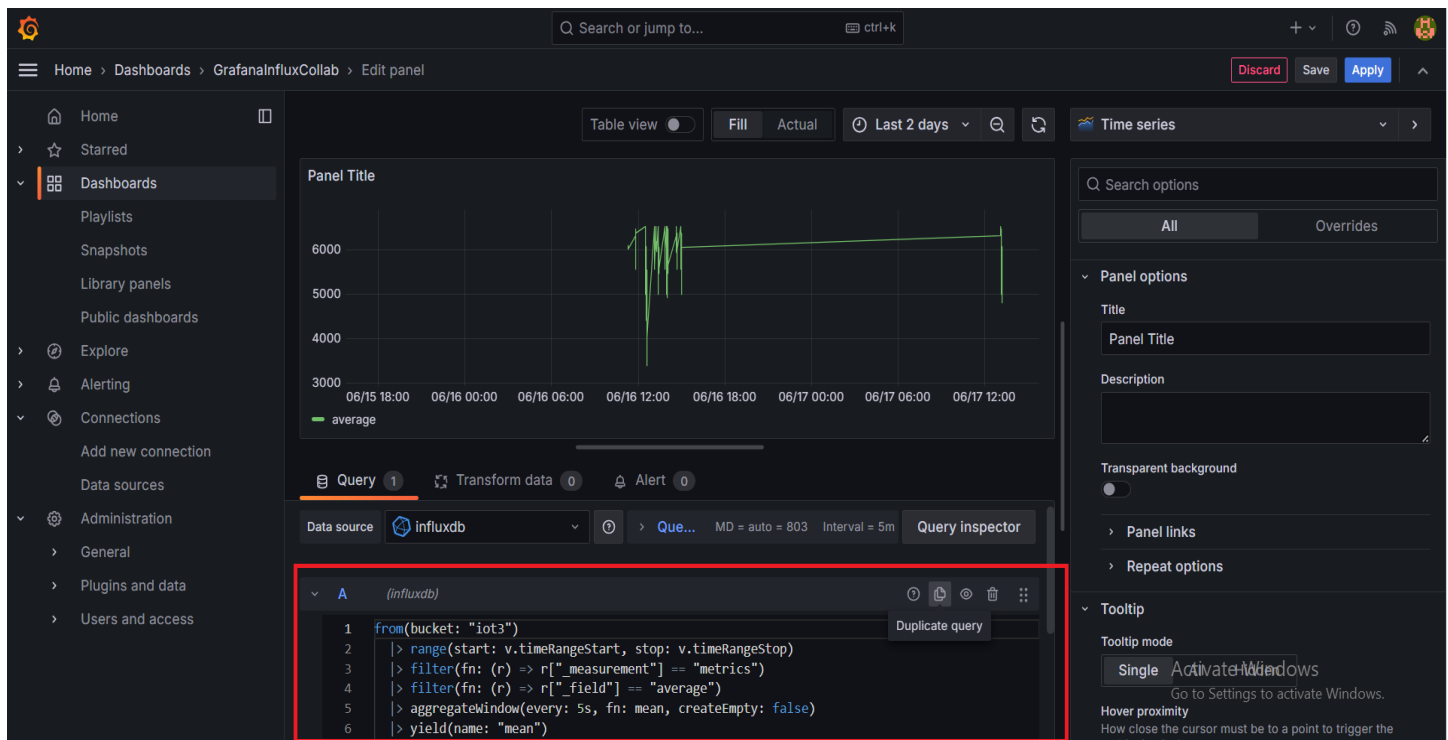
Potrebno je izabrati InfluxDB nakon čega će se otvoriti forma za konfiguraciju :



Ovde je potrebno uneti sve one podatke koje smo uneli kod prijavljivanja na InfluxDB i potrebno je uneti token koji smo generisali. Nakon toga se otvara dashboard i potrebno je uneti skriptu iz InfluxDB-a kako bi se izvršili vizuelizacija podataka u Grafani. Potrebno je kopirati sledeću skriptu koja se dobija klikom na dugme Script Editor :



Sada je potrebno kopirati ovu skriptu u Grafana dashboarda na sledeći način :



Skriptu unosimo u delu koji je označen crvenim kvadratom i dobijamo podatke koji su vizuelizovani na sledeći način.

Izgled senzor servisa i citanje podataka iz csv fajla :

```
5 async function readCitiesFromCSV(filePath) {
6   return new Promise((resolve, reject) => {
7     const cities = [];
8     fs.createReadStream(filePath)
9       .pipe(csv())
10      .on("data", (data) => {
11        console.log("Raw data:", data);
12
13        try {
14          const city = {
15            Id: data["Id"] ? parseInt(data["Id"].trim(), 10) : null,
16            City: data["City"] ? data["City"].trim() : null,
17            Country: data["Country"] ? data["Country"].trim() : null,
18            Smart_Mobility: data["Smart_Mobility "]
19              ? parseFloat(data["Smart_Mobility "].trim())
20              : null,
21            Smart_Environment: data["Smart_Environment"]
22              ? parseFloat(data["Smart_Environment"].trim())
23              : null,
24            Smart_Government: data["Smart_Government "]
25              ? parseFloat(data["Smart_Government "].trim())
26              : null,
27            Smart_Economy: data["Smart_Economy "]
28              ? parseFloat(data["Smart_Economy "].trim())
29              : null,
30            Smart_People: data["Smart_People"]
31              ? parseFloat(data["Smart_People"].trim())
32              : null,
33            Smart_Living: data["Smart_Living"]
34              ? parseFloat(data["Smart_Living"].trim())
35              : null,
36            SmartCity_Index: data["SmartCity_Index"]
37              ? parseFloat(data["SmartCity_Index"].trim())
38              : null,
39            SmartCity_Index_relative_Edmonton: data[
40              "SmartCity_Index_relative_Edmonton"
41            ]
42              ? parseFloat(data["SmartCity_Index_relative_Edmonton"].trim())
43              : null,
44          };
45
46          // Loguje parsirane podatke za svaki grad
47          console.log("Parsed city data:", city);
48
49          cities.push(city);
50        } catch (error) {
51          console.error('Error parsing data: ${error.message}');
52        }
53      })
54      .on("end", () => {
55        resolve(cities);
56      })
57      .on("error", (error) => {
58        reject(error);
59      });
60    }
61  }
62 }
```

Activate Windows
Go to Settings to activate Windows.

A ovo je kod kojim se vrši slanje ovih podataka preko mqtt brokera :

```
62
63 const mqttConfig = {
64   host: "192.168.99.100",
65   port: 1883,
66   username: "admin",
67   password: "qwerty123",
68 };
69
70 async function sendCityDataOverMQTT() {
71   const filePath = "Smart_City_index_headers.csv";
72   const topic = "sensor_dummy/values";
73
74   try {
75     const cities = await readCitiesFromCSV(filePath);
76     const mqttClient = mqtt.connect(mqttConfig);
77
78     mqttClient.on("connect", async () => {
79       console.log("Connected to MQTT broker");
80
81       let cityIndex = 0;
82
83       const interval = setInterval(() => {
84         if (cityIndex < cities.length) {
85           const city = cities[cityIndex];
86           console.log("City data:", JSON.stringify(city));
87
88           const message = JSON.stringify(city);
89           mqttClient.publish(topic, message);
90           console.log(`Published data to topic: ${topic}`);
91           cityIndex++;
92         } else {
93           clearInterval(interval);
94           mqttClient.end();
95           console.log("Finished publishing all city data");
96         }
97       }, 5000);
98     });
99   } catch (error) {
100     console.error("Error reading cities from CSV:", error);
101   }
102 }
103
104 sendCityDataOverMQTT();
105
```

Docker fajlovi :


Filter microservice docker fajl :

```
Analytics > Analytics > Dockerfile > ...
1 FROM mcr.microsoft.com/dotnet/sdk:7.0 as build-env
2 WORKDIR /src
3 |
4 COPY . .
5
6 RUN dotnet restore
7 RUN dotnet publish -c Release -o /publish
8
9 FROM mcr.microsoft.com/dotnet/aspnet:7.0 as runtime
10 WORKDIR /publish
11 COPY --from=build-env /publish .
12 ENTRYPOINT [ "dotnet", "Analytics.dll" ]
```

Command microservice docker fajl :

```
Command > Dockerfile > ...
1 FROM node:14
2
3 WORKDIR /app
4
5 COPY package*.json ./
6
7 RUN npm install
8
9 COPY . .
10
11 EXPOSE 3000
12
13 CMD ["node", "index.js"]
14 |
```

Dashboard microservice docker fajl :

Command >  Dockerfile > ...

```
1 FROM node:14
2
3 WORKDIR /app
4
5 COPY package*.json ./
6
7 RUN npm install
8
9 COPY . .
10
11 EXPOSE 3000
12
13 CMD ["node", "index.js"]
14 |
```