Введение в IoT (семинары)

Этот репозиторий содержит установочные скрипты bash и документацию.

README - в данном руководстве описаны практические работы (ДЗ) по курсу "Введение в IOT".

Онлайн университет: GeekBrains

Группа: Инженер умных устройств

Студент: Ростислав Ромашин

Урок 1. Развертывание своей системы визуализации

Все описанное в разделе выполнялась на операционной системе Ubuntu 22.04.3 LTS с использованием Docker version 24.0.6 build ed223bc.

Все действия рекомендую проводить от имени суперпользователя root.

Быстрый старт

Копируем команду представленную ниже и вставляем в терминал системы.

```
wget https://github.com/allseenn/iot/raw/main/install.sh -O install.sh &&
bash install.sh
```

В результате запустится скрипт развертывания системы на основе docker контейнеров:

- mosquitto
- telegraf
- influxdb
- grafana
- node-red
- wireguard

Логин и пароль для всех служб по умолчанию одинаковый, во избежании путаницы:

- user: admin
- pass: students

ВНИМАНИЕ: Из-за требований безопасности одной из служб общий пароль должен быть не менее 8 символов. Поэтому students - во множественном числе с **s** на конце.

После установки, через веб-браузер заходим на адреса, которые будут опубликованы скриптом перед завершением работы.

Порты для служб по умолчанию:

- mosquitto 1883/tcp
- telegraf 8092/udp, 8125/udp, 8094/tcp
- influxdb 8080/tcp
- grafana 3000/tcp
- node-red 1880/tcp
- wireguard 51820/udp

Вся служебная информация: логины, пароли, адреса и порты, токены, будет сохранена в файле ~/info.txt после завершения скрипта установки.

Описание

Предлагаю два скрипта:

- install.sh
- uninstall.sh

install.sh

Установочный bash скрипт в начале кода содержит константы, которые можно изменить по своему усмотрению:

- USERNAME - имя пользователя, одинаковое для всех служб
- PASSWORD - пароль не менее 8 символов, одинаковый для всех служб
- ORG - организация
- BUCKET - корзина
- INFLUXDB_TOKEN - токен базы InfluxDB

Вышеперечисленные константы можно оставить по умолчанию.

Константы перечисленные ниже:

- LOC_IP - локальный ip-адрес
- PUB_IP - публичный ip-адрес
- ALL_IP - маска, означающая - все адреса
- DOCKER_VERSION - требуемая версия докера, если версия ниже, то происходит автоматическое обновление до последней.

Значение последних четырех констант вычисляются автоматически, но могут быть откорректированы на свое усмотрение.

Загрузить скрипт на систему Ubuntu можно несколькими способами:

- с помощью git
- wget, curl
- copy & pase

В любом случае, после сохранения скриптов на свою систему, необходимо выставить на них бит исполнения:

```
chmod 700 *.sh
./install.sh
```

Либо запускать скрипт как аргумент интерпретатора:

```
bash install.sh
```

Первым делом скрипт install.sh проверит наличие docker и его версию, в случае отсутствия либо низкой версии программы, будет произведена установка.

Скрипт автоматически создаст в домашней директории пользователя (в случае рута в /root) структуру каталогов и в них конфигурационные файлы, которые автоматически будут примонтированы к соответствующим докер контейнерам.

В конце работы скрипта будет выведена информация о локальных, публичных адресах и о портах каждой службы.

Проброс портов через роутер в данном руководстве не рассматривается, предполагается, что вы знакомы с курсом "Компьютерные сети".

Так же будет создано несколько конфигураций для ВПН WireGuard. И в терминал будет выведен QR-код клиента.

Информация по настройке и установке WireGuard клиента не рассматривается, а может быть получена с оффсайта.

```
...готовлю список подключениий для внутренних адресов...
..готовлю инфу по внешним адресам...
{"datasource":{"id":1,"uid":"d32c82bc-7f49-40c6-8b17-280026175ed6","orgId":1,"name":"InfluxDB","type":"infl
ccess":"proxy","url":"http://192.168.1.8:8086","user":"","database":"","basicAuth":false,"basicAuthUser":""
":"Bearer","organization":"IoT","version":"Flux"},"secureJsonFields":{"token":true},"version":1,"readOnly":
ние от графаны до инфлюкса...
...сохраняю параметры подключения в файл ~/info.txt...
ПАРАМЕТРЫ ДЛЯ ЛОКАЛЬНОГО ПОДКЛЮЧЕНИЯ
Grafana http://192.168.1.8:3000
NodeRed http://192.168.1.8:1880
Influx2 http://192.168.1.8:8086

ПАРАМЕТРЫ ДЛЯ ПОДКЛЮЧЕНИЯ ИЗВНЕ, ЕСЛИ ПРОБРОШЕНЫ ПОРТЫ
Grafana http://46.188.107.132:3000
NodeRed http://46.188.107.132:1880
Influx2 http://46.188.107.132:8086

ПАРАМЕТРЫ ДЛЯ АВТОРИЗАЦИИ
username: admin
password: students
внешний ip: 46.188.107.132
локальный ip: 192.168.1.8
token influxdb: kFhczFje8dRm2SXK1V9Ds7xpcJTr6wVUS881KQoUQWE-QAfcg-S-6j1FvFiSvWW0wTPlmWHCvXf_JU1hRx5rZg==
Версия WireGuard для андроида: https://play.google.com/store/search?q=wireguard
Конфиги настроки WireGuard лежат ~/wireguard/config/wg_confs/wg0.conf
...готовлю код для WireGuard...
QR-КОД ДЛЯ НАСТРОЙКИ WIREGUARD
```

На рисунке показан пример финального вывода служебной информации установочного скрипта install.sh

В результате работы install.sh будет произведен деплой полностью рабочей системы. С настроенными паролями для каждой службы.

Более того, с помощью API графаны мне удалось создать datasource источник подключения к influxdb.

Скорее всего всю работу проведенную в первом занятии по настройке и визуализации можно сделать с помощью API служб и их команд. Но, это следующий этап)

uninstall.sh

Скрипт деинсталляции, позволяет полностью удалить развернутые контейнеры их конфиги и виртуальные сетевые интерфейсы.

В процессе экспериментов и настройки установочного скрипта, для ускорения процесса разработки использовал команду:

```
./uninstall.sh && ./install.sh
```

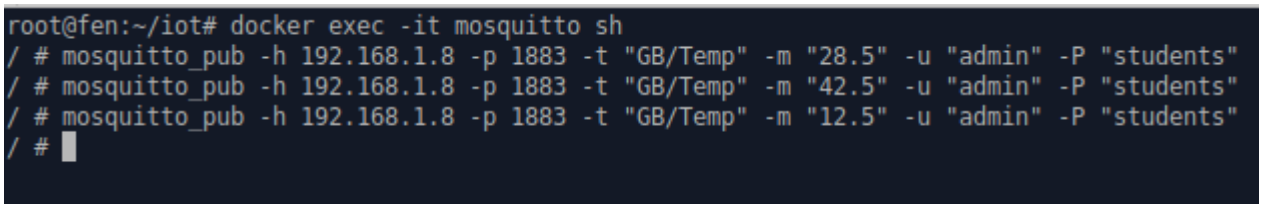
Это позволяет одной строкой удалить старый набор контейнеров и установить новый измененный.

Скрины

Mosquitto

Зайдя в shell контейнера mosquitto, можно отправлять mqtt события из терминала.

```
$ docker exec -it mosquitto sh
/ # mosquitto_pub -h 192.168.1.8 -p 1883 -t "GB/Temp" -m "28.5" -u "admin" -
P "students"
```



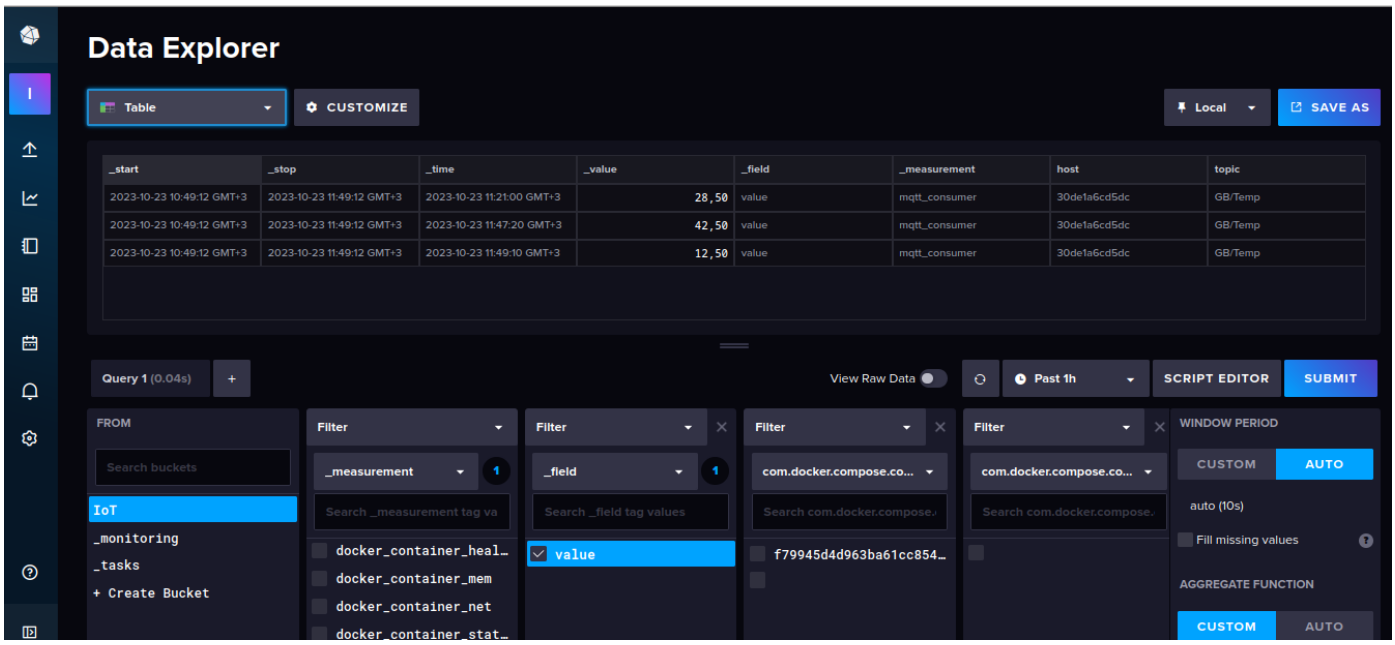
Telegraf

Телеграф прослушивает заданные в его конфиге топики и записывает события в базу InfluxDB:

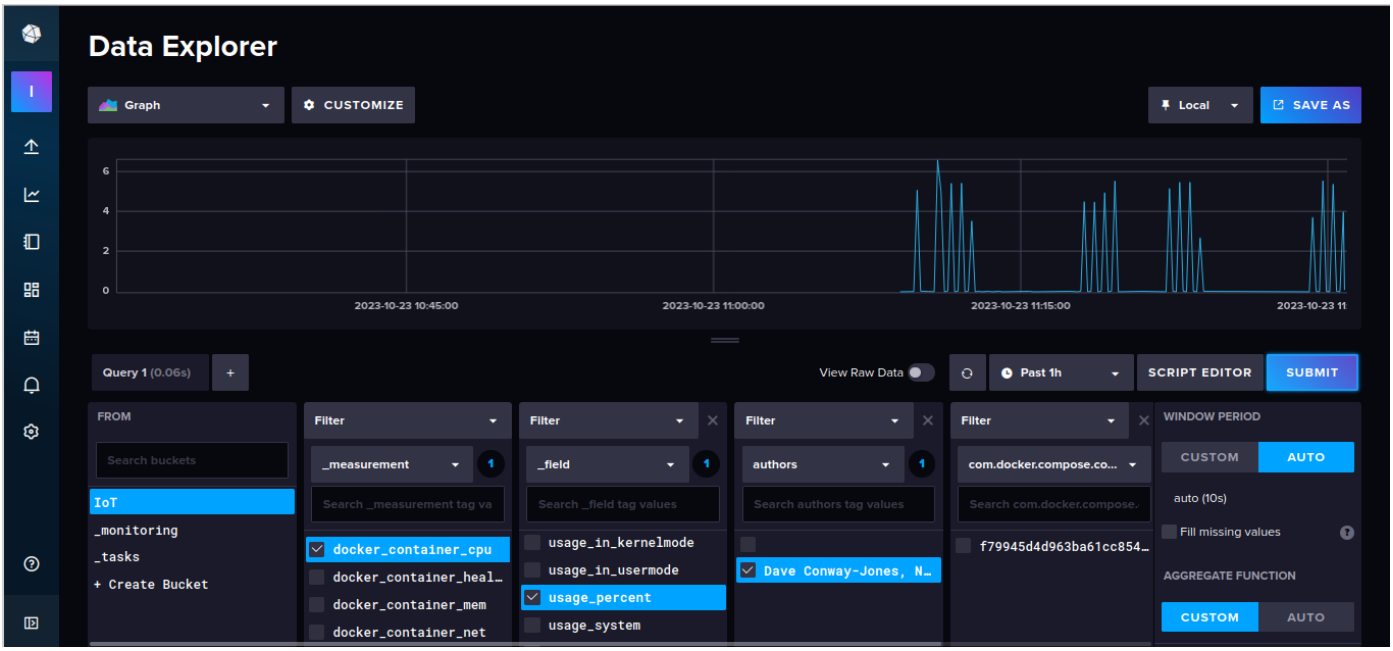
```
[agent]
 interval = "3s"
 round_interval = true
 metric_batch_size = 1000
 metric_buffer_limit = 10000
 collection_jitter = "0s"
 flush_interval = "3s"
 flush_jitter = "0s"
 precision = ""
 hostname = ""
 omit_hostname = false

[[outputs.influxdb_v2]]
  urls = ["http://192.168.1.8:8086"]
  token = "kFhczFje8dRm2SXK1V9Ds7xpcJTr6wVUS88
  organization = "IoT"
  bucket = "IoT"

[[inputs.mqtt_consumer]]
  servers = ["tcp://192.168.1.8:1883"]
  topics = ["#"]
  username = "admin"
  password = "students"
  data_format = "value"
  data_type = "float"

[[inputs.docker]]
  endpoint = "unix:///var/run/docker.sock"
```

InfluxDB

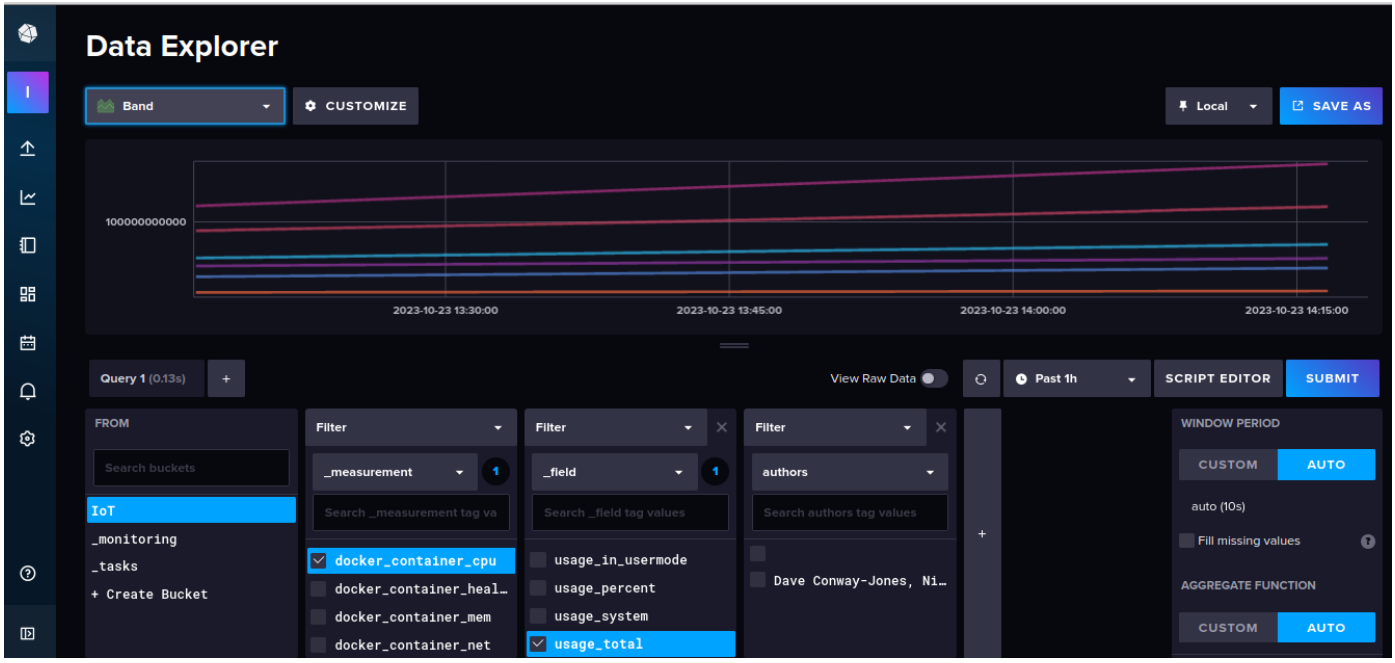
После того, как мы послали через терминал mosquitto несколько событий, телеграф их записал в базу, зайдя в вебку InfluxDB, наблюдаем по нашим топикам наглядную информацию:



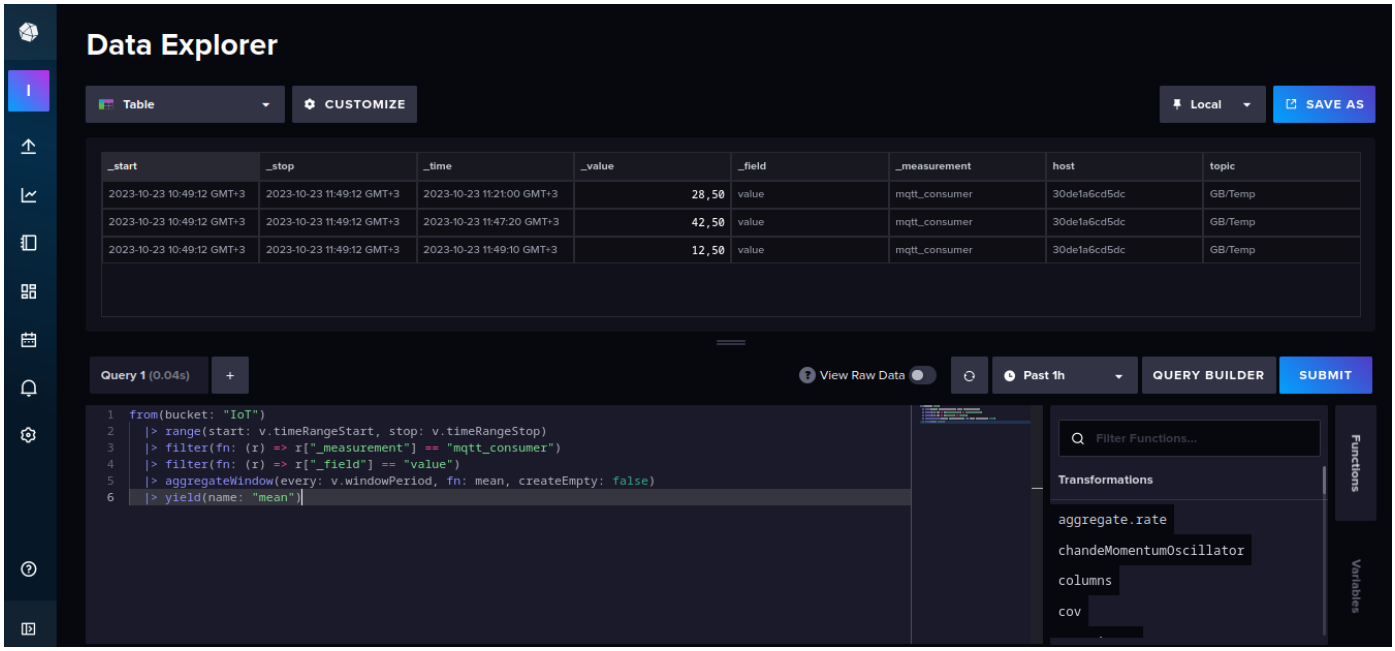
Информацию можно визуализировать различными способами, такими как графики, спидометры или гистограммы:



Обратил внимание, что докер контейнеры, благодаря правильной настройке, также скидывают свои метрики в базу, так можно наблюдать общую загрузку процессора у каждого контейнера:

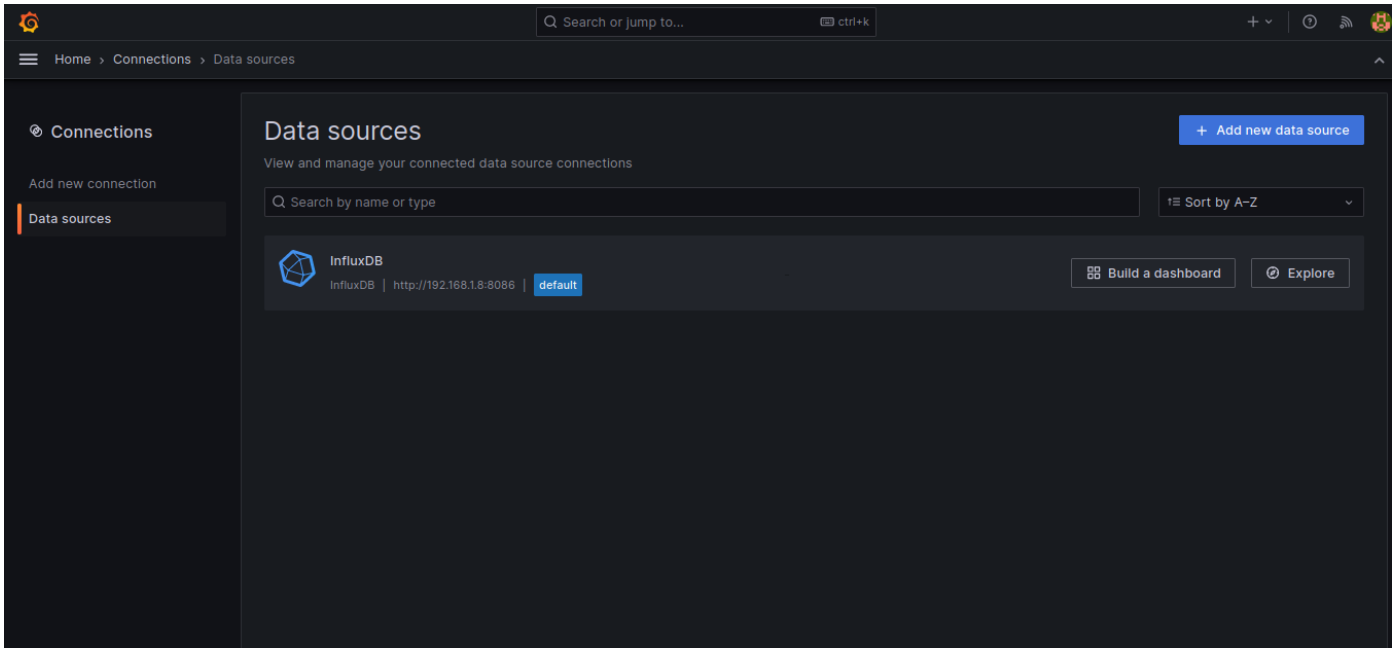


Запрос influxdb можно скопировать для последующего отображения в grafana:

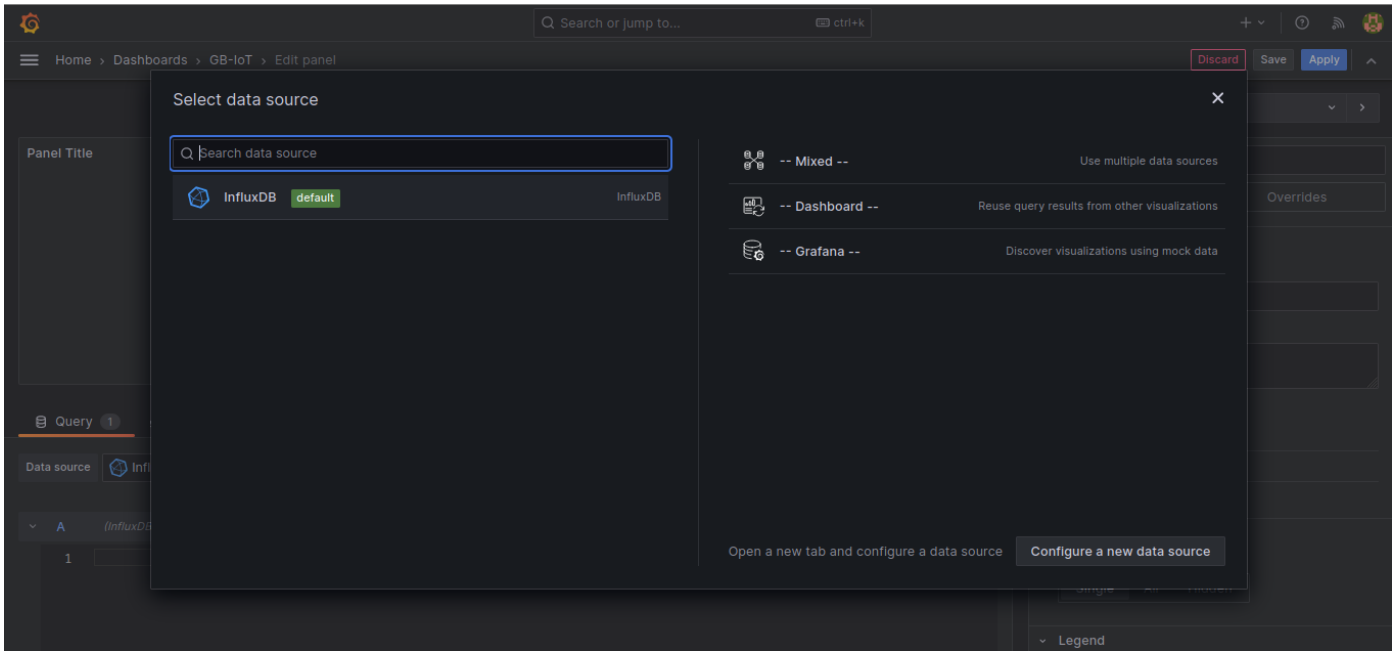


Grafana

Благодаря API Grafana мы автоматически создали подключение к базе InfluxDB и оно уже присутствует в нашей системе:

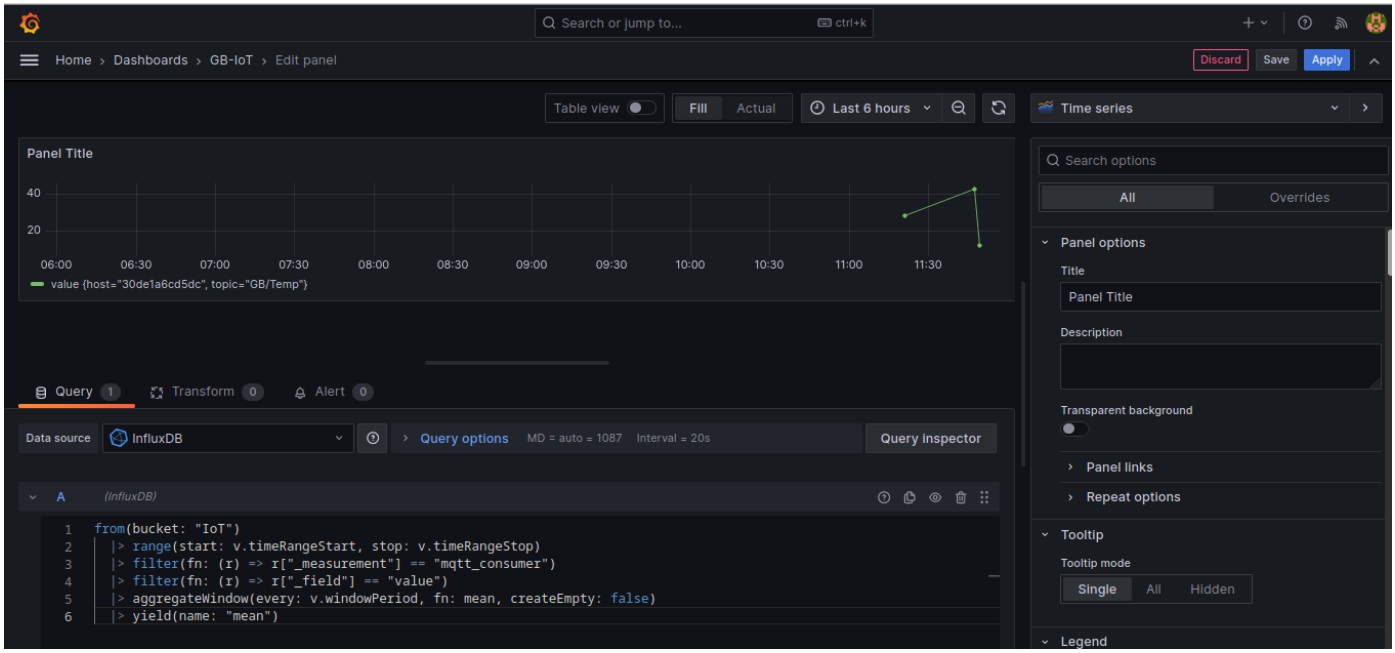


Остается только выбрать подключенный источник данных:



И вставить, скопированный ранее, из InfluxDB запрос:

```
from(bucket: "IoT")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "mqtt_consumer")
  |> filter(fn: (r) => r["_field"] == "value")
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> yield(name: "mean")
```

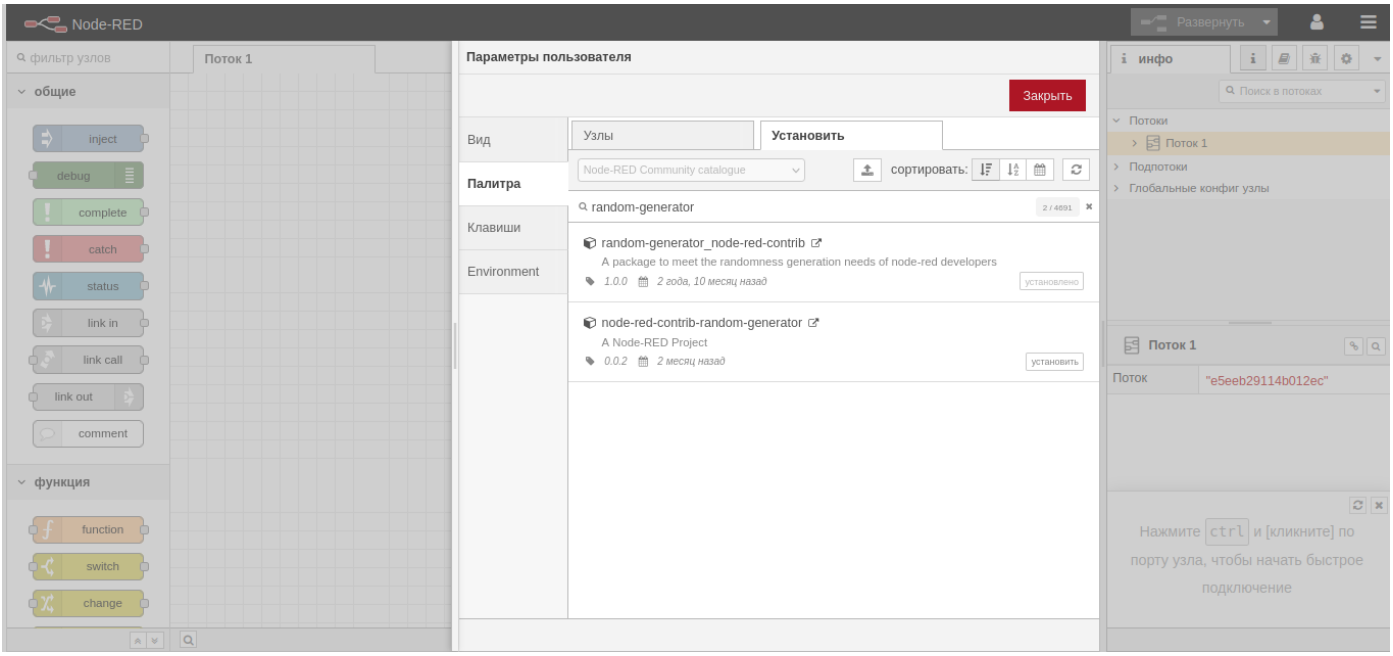


Отобразится графическая информация сгенерированная с помощью mosquitto, но средствами Grafana, вид представления можно поменять на спидометр, таблицу и др.:

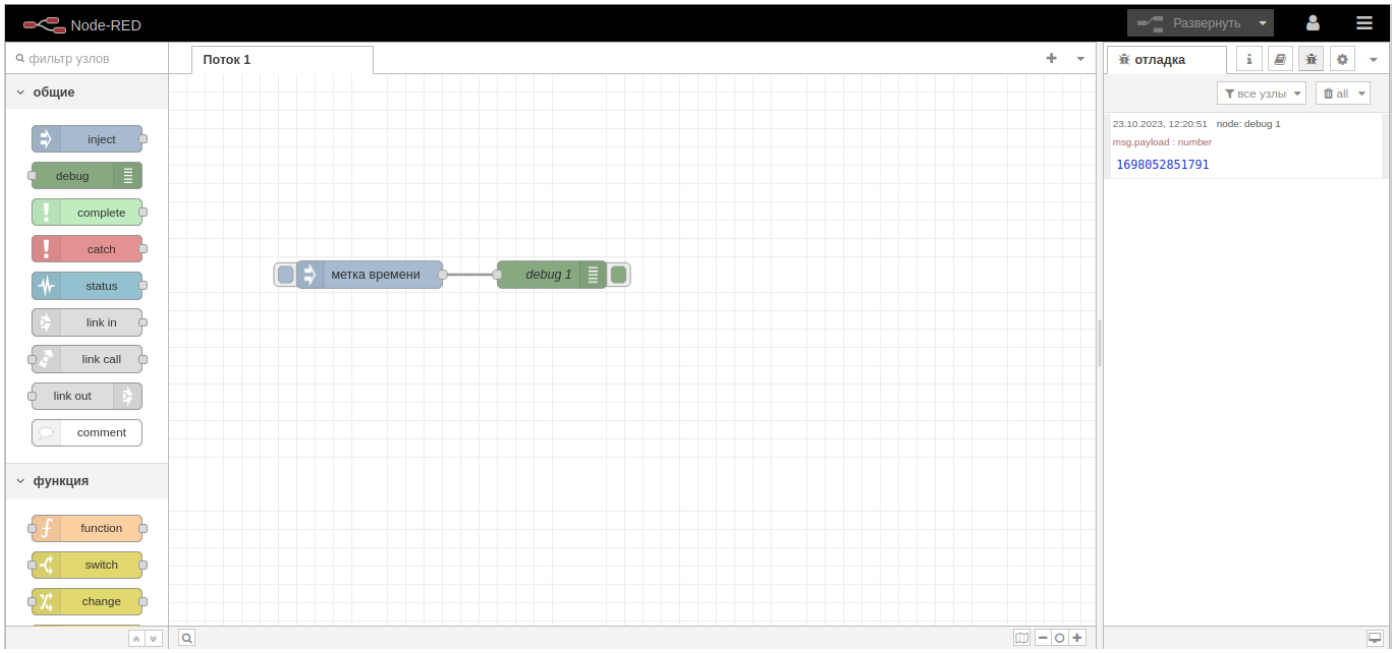
Сохранив данное представление, получим готовый дашборд.

Node-red

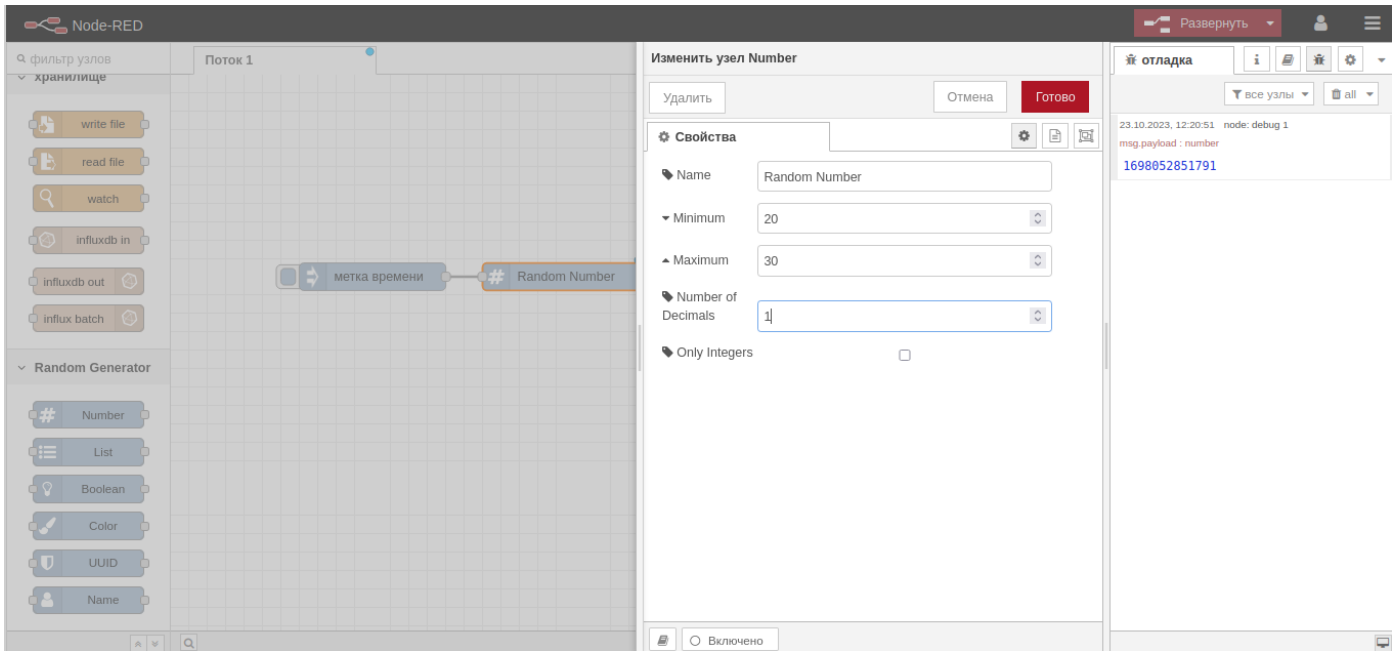
Установим палитру для случайной генерации чисел:



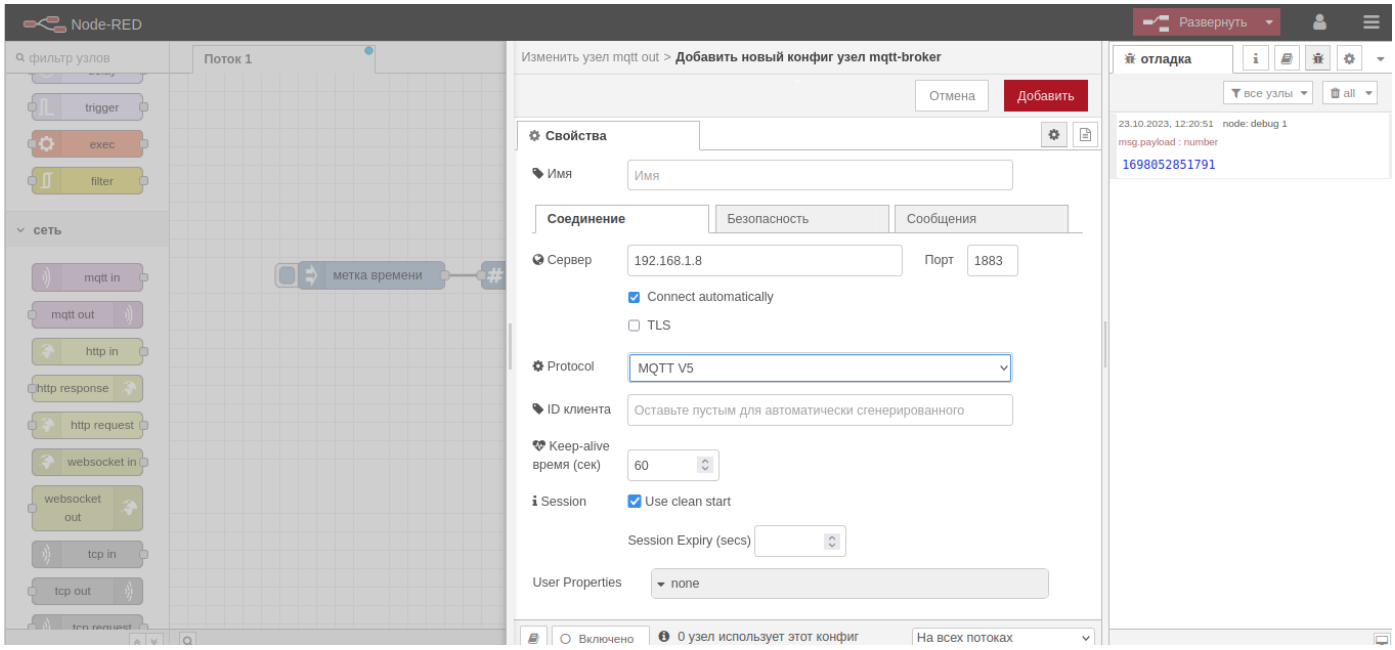
Подключив элемент отладки при нажатии на кнопку метки времени, в правой панели наблюдаем порядковый номер действия:



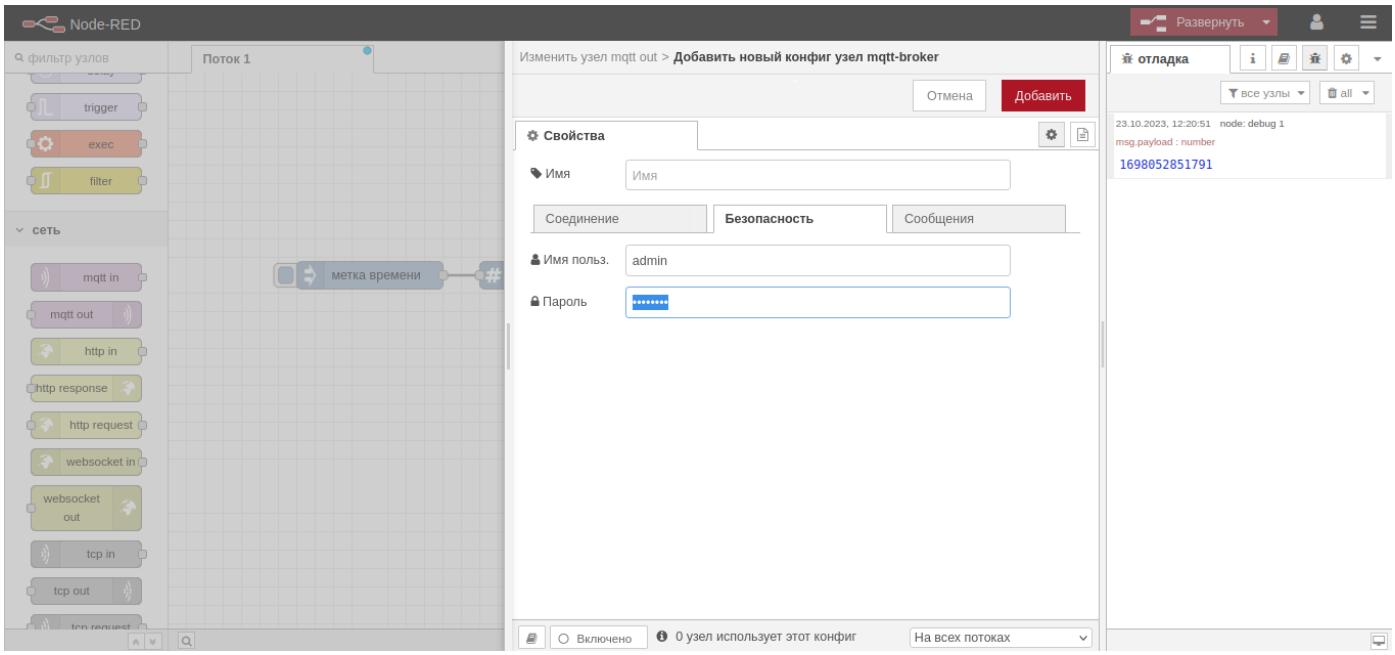
Добавим элемент Random Number и зададим его параметры:



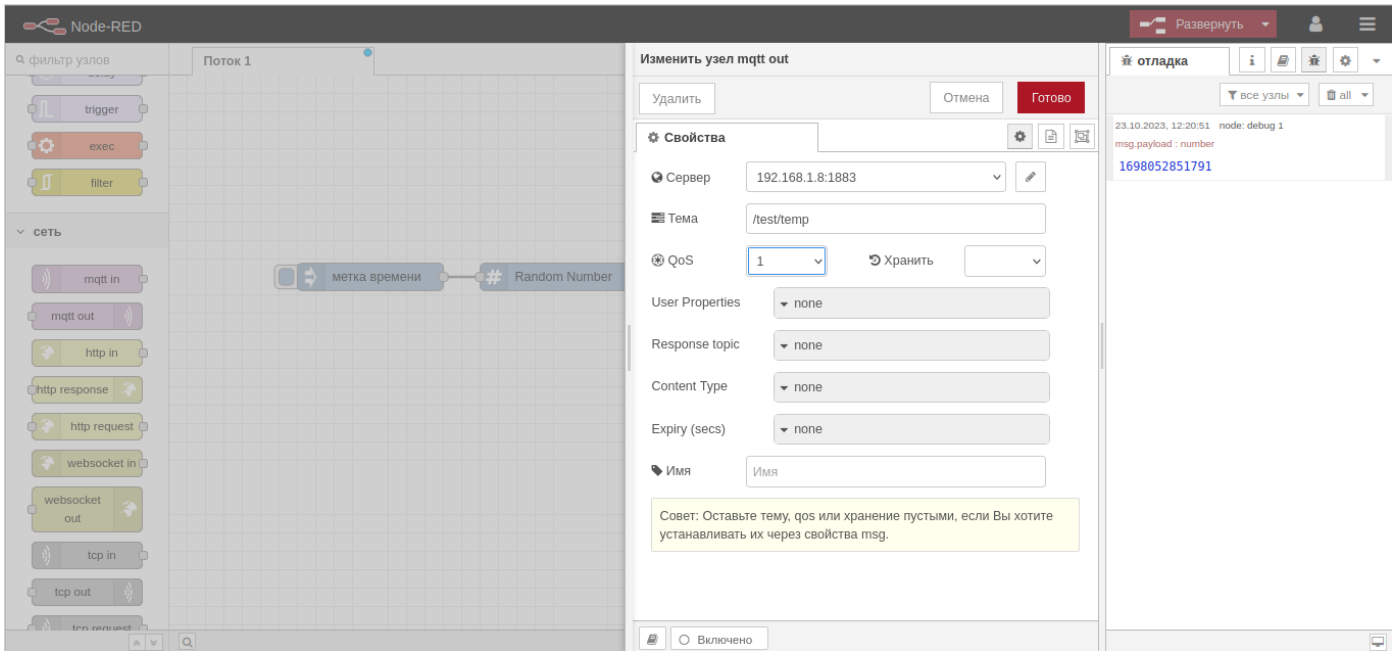
Добавим элемент mqtt-out и настроим его параметры, с помощью двойного клика по нашему элементу:



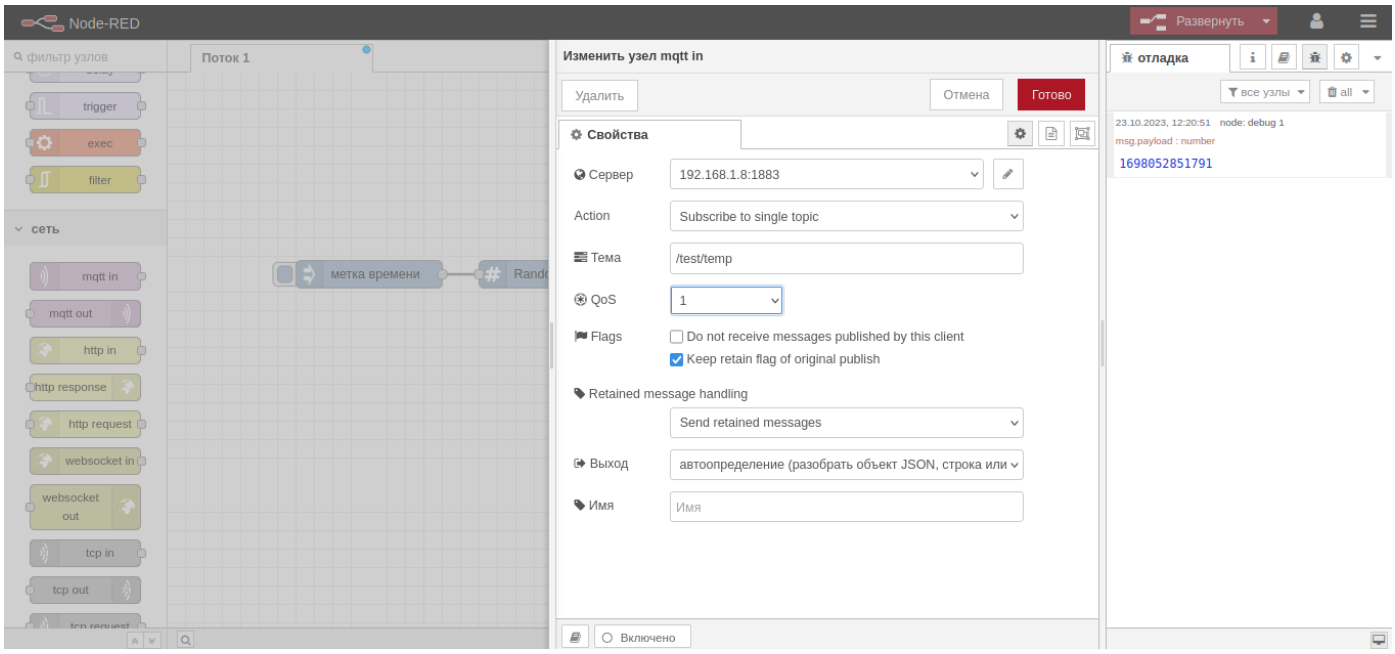
Для элемента mqqt-out введем параметры безопасности:



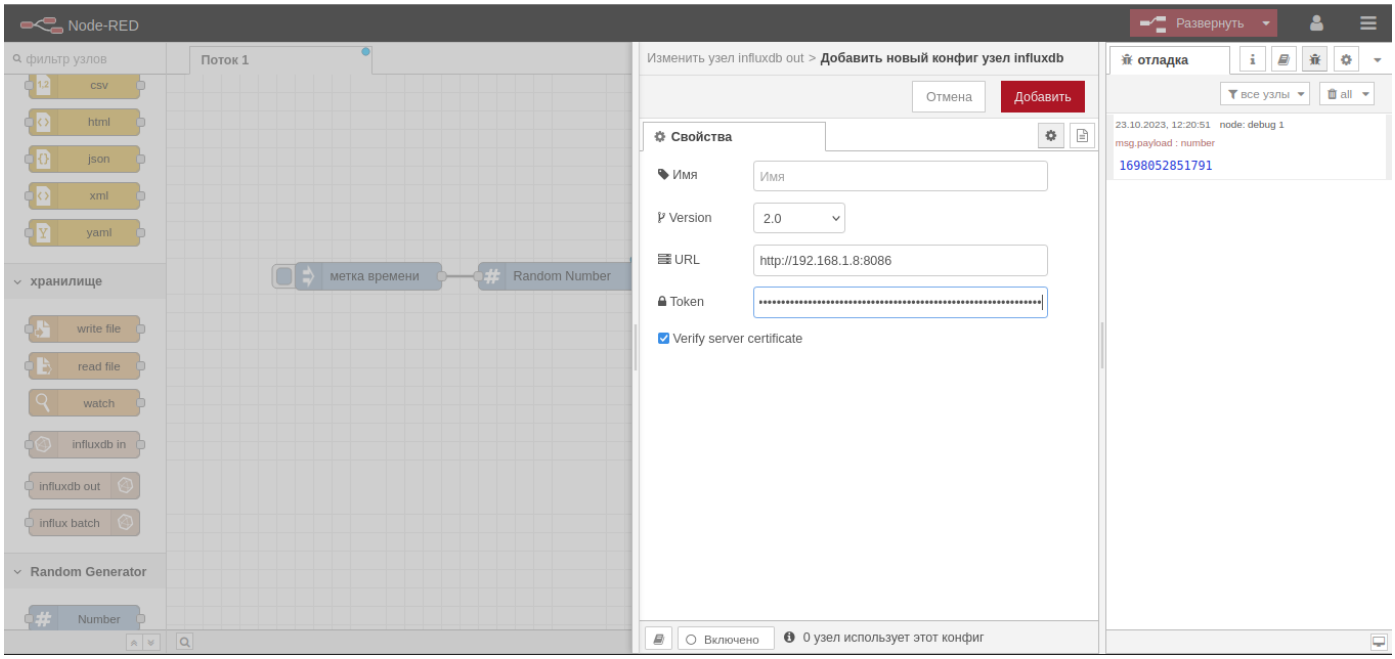
Настроим для mqtt-out тему и QoS:



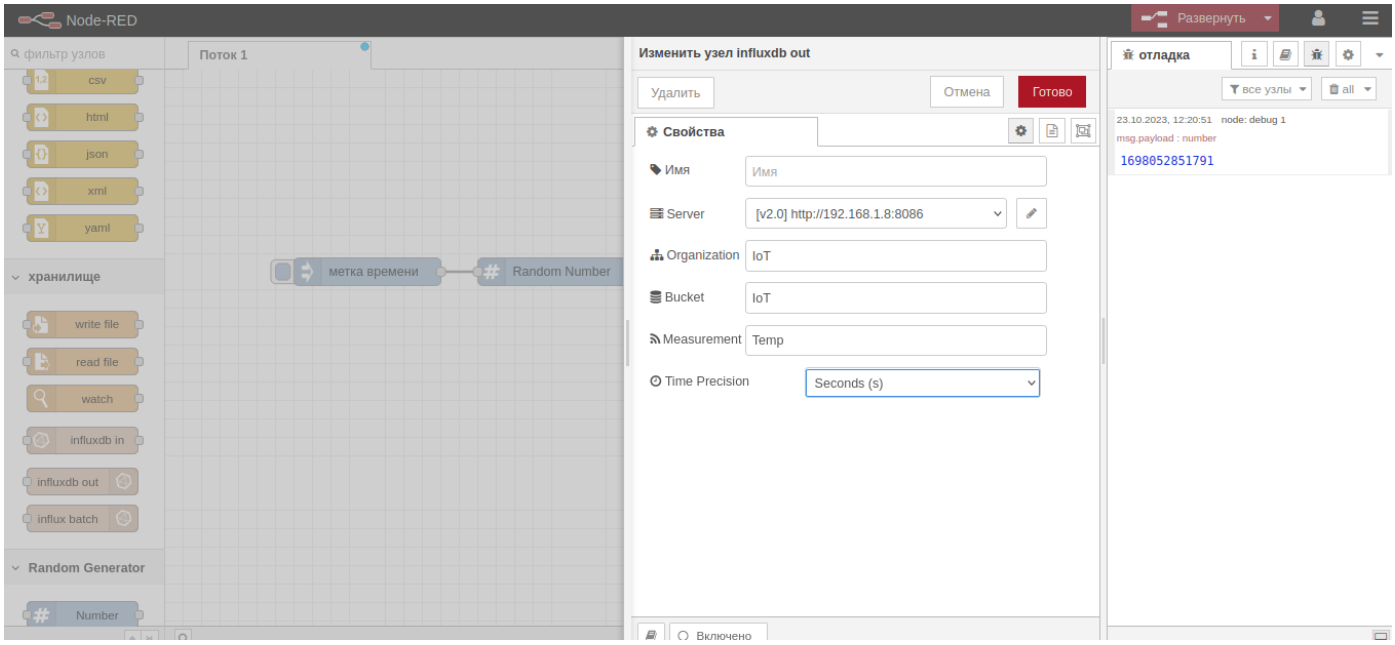
Добавив элемент mqtt-in, настроим его параметры:



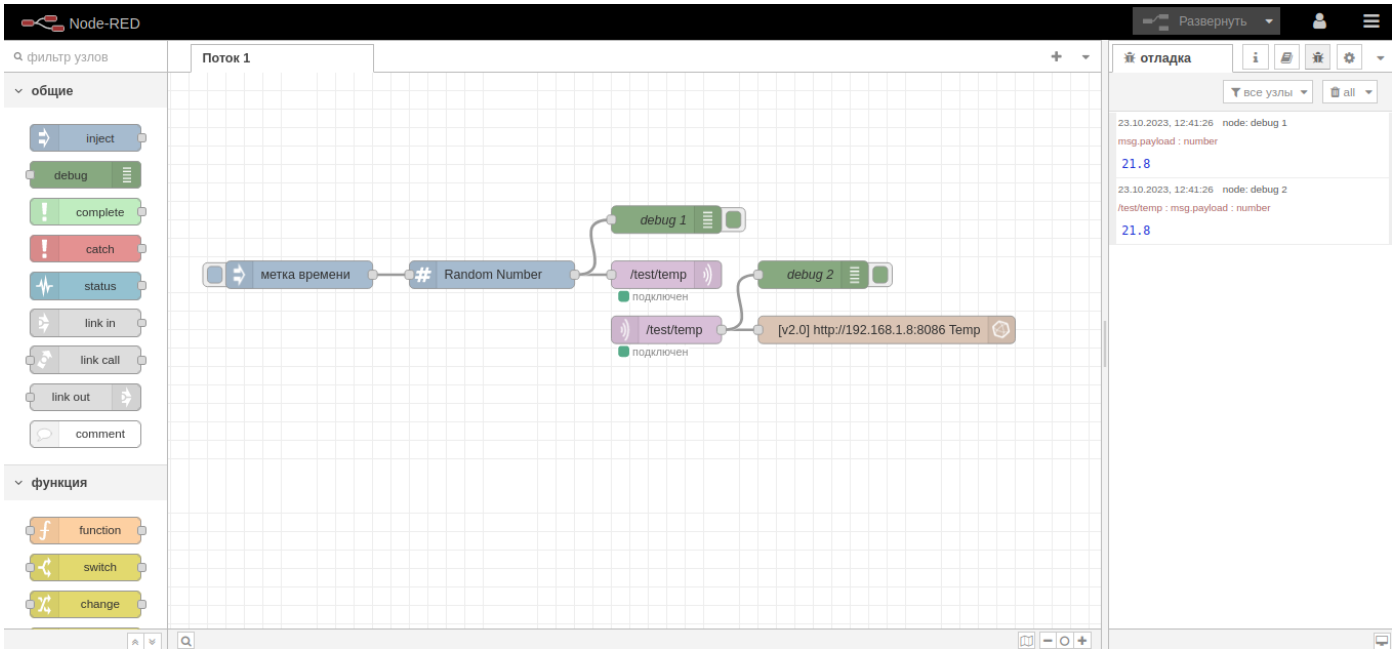
Настроим хранилище InfluxDB out:



Зададим параметры для Organization и Bucket:

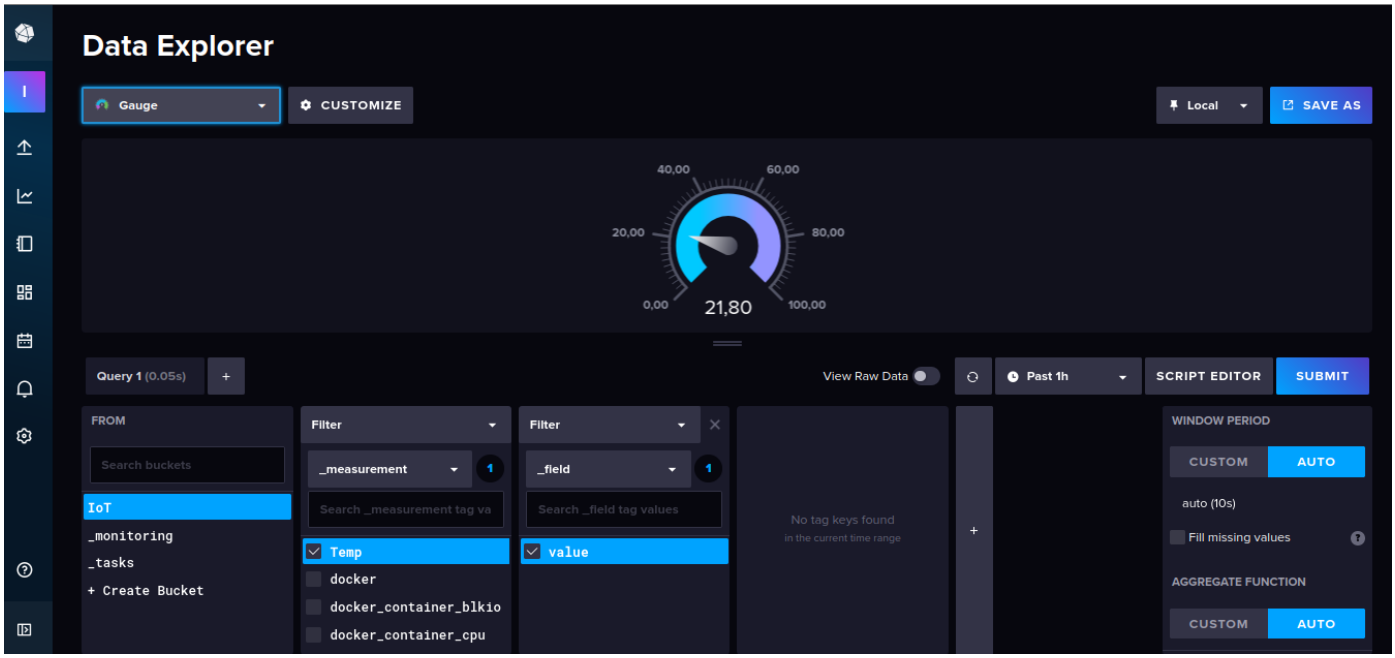


После нажатия кнопки "Развернуть" (Deploy), сетевые элементы входа и выхода изменили статус на "подключено":

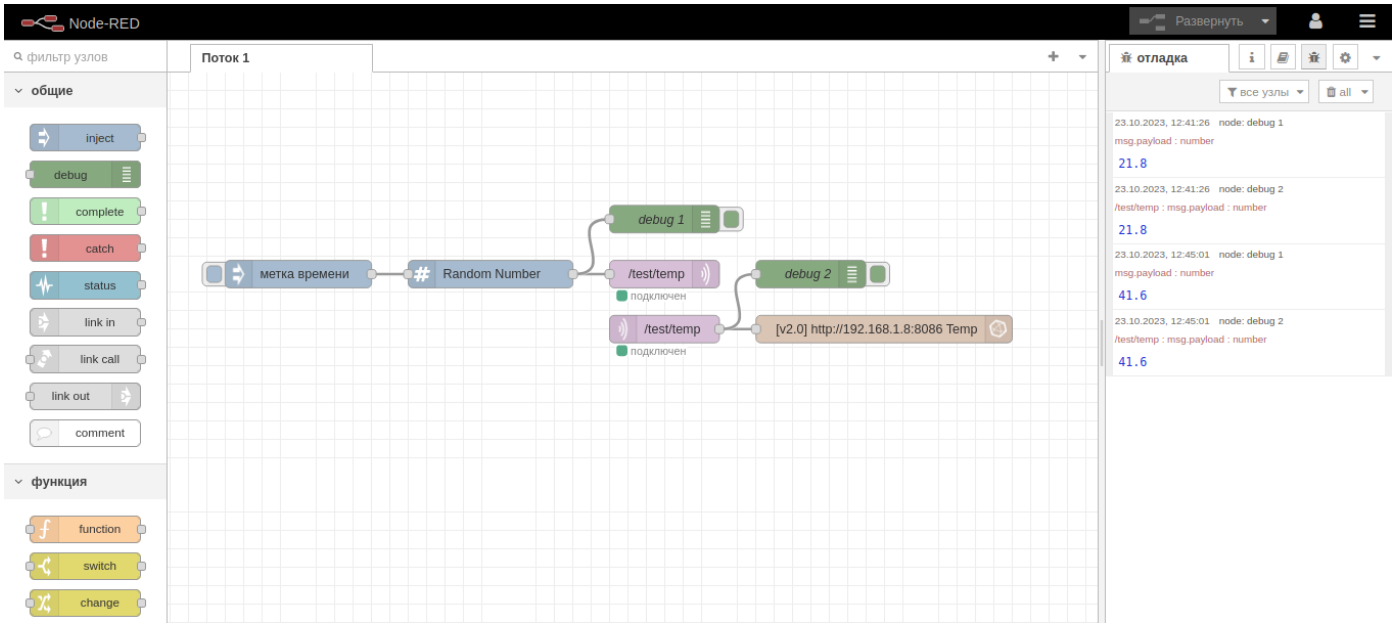


Заключение

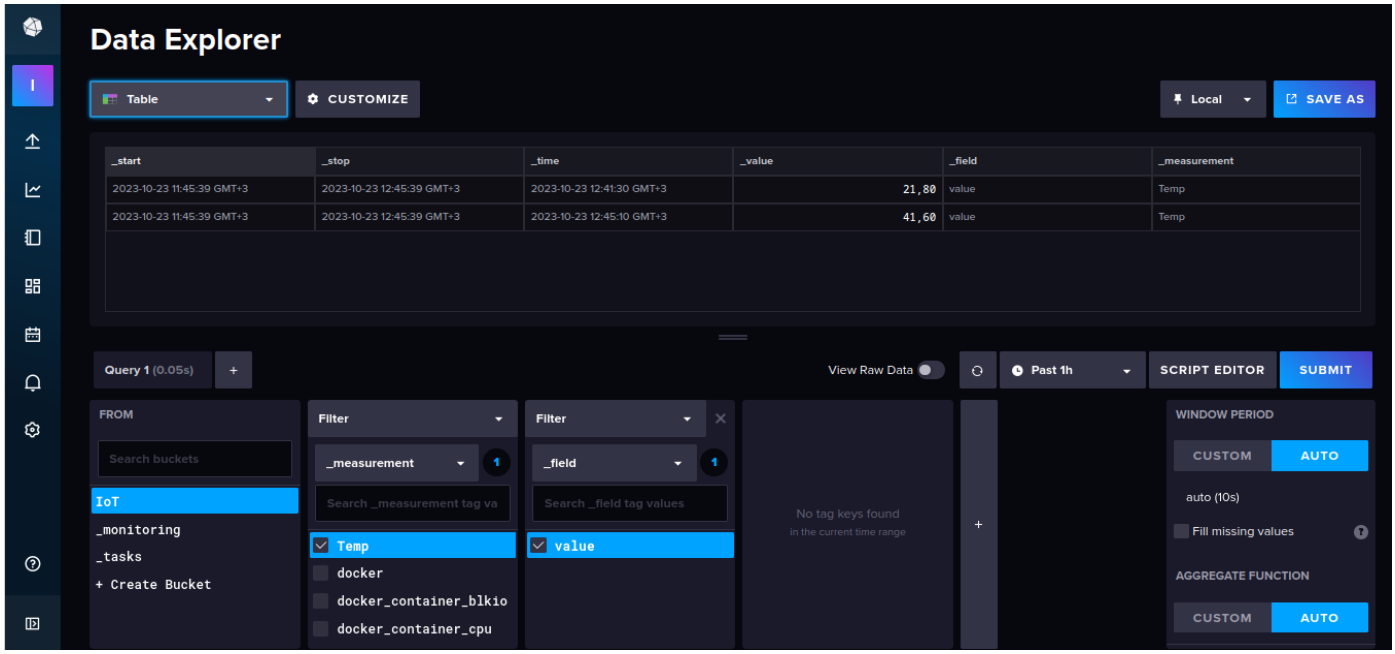
В результате произведенных действий в Node-Red, мы видим новые данные в базе InfluxDB:



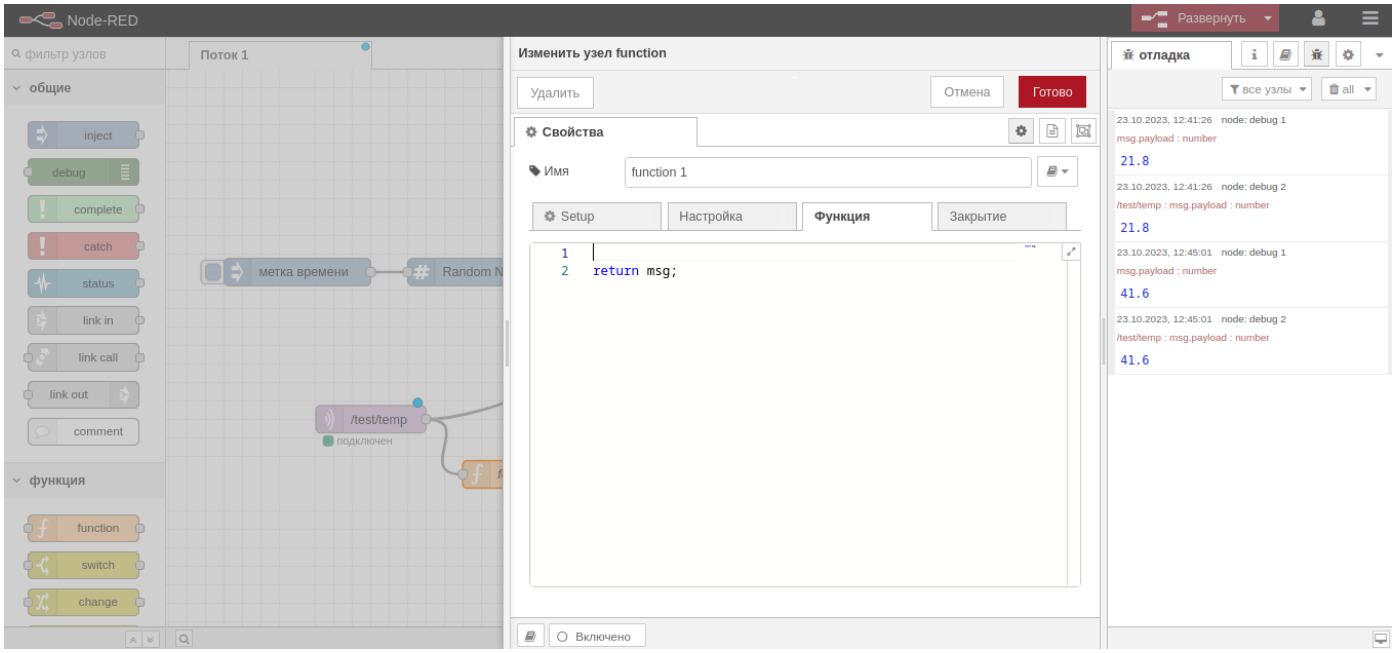
Сгенерируем больше случайных событий в Node-Red:



Все эти действия зафиксированы в базе:



Node-Red позволяет в разрез элементов вставлять функции написанные на JavaScript:



А установка новых палитр позволяет расширить функционал и набор поддерживаемого оборудования.

Подключил палитру node-red-node-serailport, с помощью нее попробую подключить устройство, купленный ранее термодатчик, для выполнения второго задания.