# Описание

Работа проводилась в системе **ubuntu** *24.04*, установленной на стареньком ПК.

ПК имеет локальный адрес *192.168.100.100* и локального пользователя **alexs**

Для начало установил в систему **docker** и **docker-compose**:

apt-get -y install git docker-ce docker-ce-cli containerd.io \

docker-buildx-plugin docker-compose-plugin

Позже, с помощью докер файла будут установлены образы:

* mosquitto
* influxdb2
* telegraf
* grafana
* node-red
* wide-guard

Чтобы избежать путаницы, на всех контейнерах, с помощью конфигов, будет задан пользователь **admin** и пароль **alexs1234**

# Конфиги

Далее, создал и разместил конфигурационные файлы в корне домашней директории своего пользователя **alexs** на линуксе

## Mosqutto

Создал 3 директории

mkdir -p ~/mosquitto/config

mkdir -p ~/mosquitto/data

mkdir -p ~/mosquitto/log

1 конфиг *~/mosquitto/config/mosquitto.conf*

listener 1883

allow\_anonymous true

listener 8081

protocol websockets

persistence true

persistence\_location /mosquitto/data/

password\_file /mosquitto/config/password.txt

log\_dest file /mosquitto/log/mosquitto.log

1 файл паролей *~/mosquitto/config/password.txt*

admin:\$7\$101\$teVvE02cFftoHzjK\$1QJTtgILlvnJzEa1JQswCZMJKIQluIIqtpxtPaBohau6apYviWacbtJACae+nkEZyNkVwDKXZucUIPw3QqxLAw==

## InfluxDB2

2 директории

mkdir -p ~/influxdb2/data

mkdir -p ~/influxdb2/conf

## Telegraf

1 директория

mkdir -p ~/telegraf/conf

1 конфиг *~/telegraf/conf/telegraf.conf*

[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 = "telegraf"

omit\_hostname = true

[[outputs.influxdb\_v2]]

urls = ["http://192.168.100.100:8086"]

token = "P5AgAduLX1t8O5ZHvlykrGgbwKyiY2mlqYOTq5QFFVmWfIIVsiaXLbRy\_fa77j0JV5JChNsoVgqoNanXHw3EQg=="

organization = "IoT"

bucket = "IoT"

[[inputs.mqtt\_consumer]]

servers = ["tcp://192.168.100.100:1883"]

topics = ["#"]

username = "admin"

password = "alexs1234"

data\_format = "value"

data\_type = "float"

[[inputs.docker]]

endpoint = "unix:///var/run/docker.sock"

## Grafana

mkdir -p ~/grafana/data

mkdir -p ~/grafana/conf

mkdir -p ~/grafana/log

Создал конфиг *~/grafana/conf/grafana.ini*

[server]

http\_addr = 0.0.0.0

protocol = http

http\_port = 3000

[log]

level = debug

[security]

admin\_user = admin

admin\_password = alexs1234

[datasources]

query\_language = flux

name = InfluxDB

url = http://192.168.100.100:8086

token = P5AgAduLX1t8O5ZHvlykrGgbwKyiY2mlqYOTq5QFFVmWfIIVsiaXLbRy\_fa77j0JV5JChNsoVgqoNanXHw3EQg==

organization = IoT

bucket = IoT

## Node-Red

1 Директория

mkdir -p ~/node-red/data

1 Конфиг *~/node-red/data/settings.js*

module.exports = {

flowFile: 'flows.json',

flowFilePretty: true,

adminAuth: {

type: "credentials",

users: [{

username: "admin",

password: "$2y$05$zTebD8did4dpUlW2pKFkTe3eNFJzurRHj9WRbp4X0YNwOuNChCdxe",

permissions: "\*"

}]

},

uiPort: process.env.PORT || 1880,

diagnostics: {

enabled: true,

ui: true,

},

runtimeState: {

enabled: false,

ui: false,

},

logging: {

console: {

level: "info",

metrics: false,

audit: false

}

},

exportGlobalContextKeys: false,

externalModules: {

},

editorTheme: {

palette: {

},

projects: {

enabled: false,

workflow: {

mode: "manual"

}

},

codeEditor: {

lib: "monaco",

options: {

}

},

markdownEditor: {

mermaid: {

enabled: true

}

},

},

functionExternalModules: true,

functionTimeout: 0,

functionGlobalContext: {

},

debugMaxLength: 1000,

mqttReconnectTime: 15000,

serialReconnectTime: 15000,

}

## WireGurard

1 директория

mkdir -p ~/wireguard/config

# Докер

Чтобы была возможность обращаться к образам не только с убунту, но и с любого компьютера в локальной сети, необходимо создать мост

docker network create bridge

# docker-compose

Создал файл в корне домашней директории **~/docker-compose.yml**

services:

influxdb:

container\_name: influxdb

image: influxdb:latest

environment:

- TZ=Europe/Moscow

- DOCKER\_INFLUXDB\_INIT\_MODE=setup

- DOCKER\_INFLUXDB\_INIT\_USERNAME=admin

- DOCKER\_INFLUXDB\_INIT\_PASSWORD=alexs1234

- DOCKER\_INFLUXDB\_INIT\_ORG=IoT

- DOCKER\_INFLUXDB\_INIT\_BUCKET=IoT

- DOCKER\_INFLUXDB\_INIT\_ADMIN\_TOKEN=P5AgAduLX1t8O5ZHvlykrGgbwKyiY2mlqYOTq5QFFVmWfIIVsiaXLbRy\_fa77j0JV5JChNsoVgqoNanXHw3EQg==

ports:

- "8086:8086"

volumes:

- ~/influxdb2/data:/var/lib/influxdb2

- ~/influxdb2/conf:/etc/influxdb2/

networks:

- influxdb-net

restart: always

wireguard:

container\_name: wireguard

image: lscr.io/linuxserver/wireguard:latest

cap\_add:

- NET\_ADMIN

- SYS\_MODULE

environment:

- PUID=1000

- PGID=1000

- TZ=Europe/Moscow

- SERVERURL=8.8.8.8

- SERVERPORT=1871

- PEERS=5

- PEERDNS=1.1.1.1

- INTERNAL\_SUBNET=10.10.10.0

- ALLOWEDIPS=0.0.0.0/0

- LOG\_CONFS=false

volumes:

- ~/wireguard/config:/config

- /lib/modules:/lib/modules

ports:

- 51820:51820/udp

sysctls:

- net.ipv4.conf.all.src\_valid\_mark=1

restart: always

telegraf:

container\_name: telegraf

image: telegraf:latest

environment:

- TZ=Europe/Moscow

volumes:

- ~/telegraf/conf/:/etc/telegraf/

- /var/run/docker.sock:/var/run/docker.sock

networks:

- telegraf-net

restart: always

grafana:

container\_name: grafana

image: grafana/grafana-oss:latest

environment:

- TZ=Europe/Moscow

ports:

- "3000:3000"

volumes:

- ~/grafana/data:/var/lib/grafana/

- ~/grafana/log:/var/log/grafana/

- ~/grafana/conf:/etc/grafana/

links:

- influxdb

networks:

- grafana-net

restart: always

mosquitto:

container\_name: mosquitto

image: eclipse-mosquitto:latest

environment:

- TZ=Europe/Moscow

volumes:

- ~/mosquitto/config:/mosquitto/config

- ~/mosquitto/data:/mosquitto/data

- ~/mosquitto/log:/mosquitto/log

ports:

- 1883:1883

- 8081:8081

networks:

- mosquitto-net

restart: always

node-red:

container\_name: node-red

image: nodered/node-red:latest

environment:

- TZ=Europe/Moscow

ports:

- "1880:1880"

volumes:

- ~/node-red/data:/data

networks:

- node-red-net

restart: always

networks:

node-red-net:

influxdb-net:

mosquitto-net:

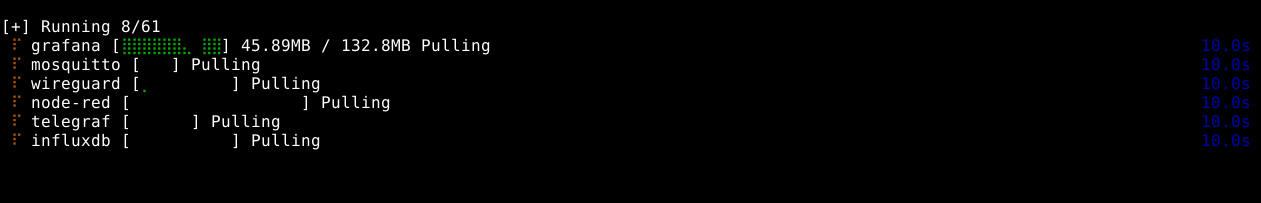
grafana-net:

telegraf-net:

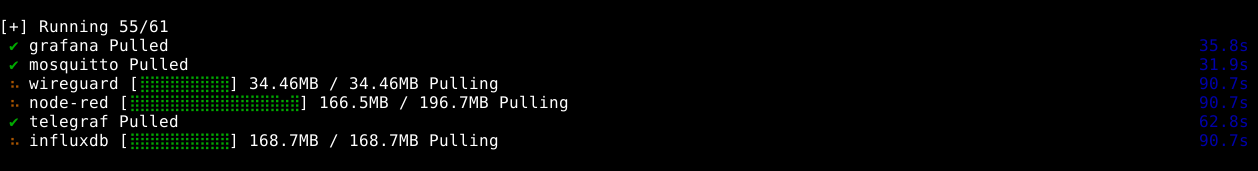
## Запуск установки контейнеров

docker compose -f ~/docker-compose.yml up -d

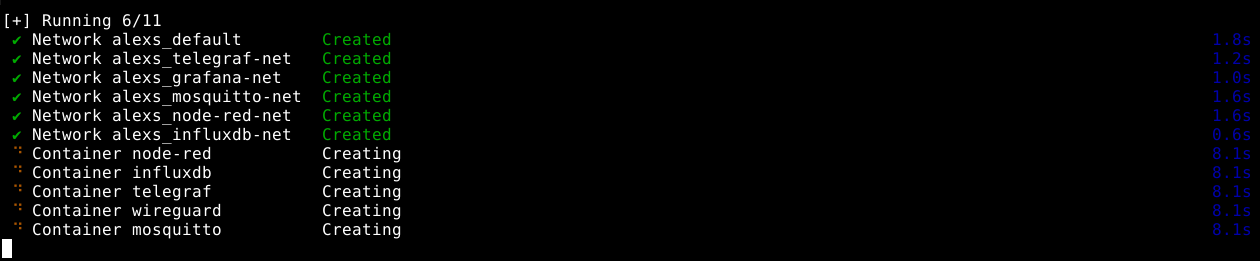
После запуска команды выше пошел процесс скачивания образов с главного регистра **docker**



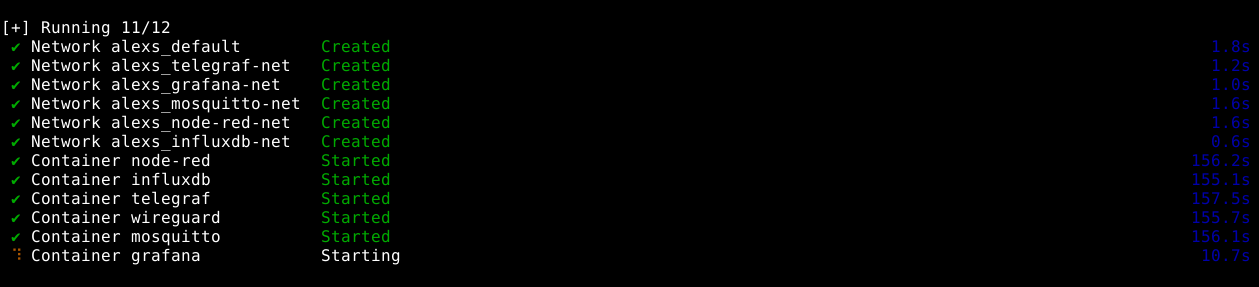
Постепенно все образы закачались



Далее создались внутренние сети для каждого образа



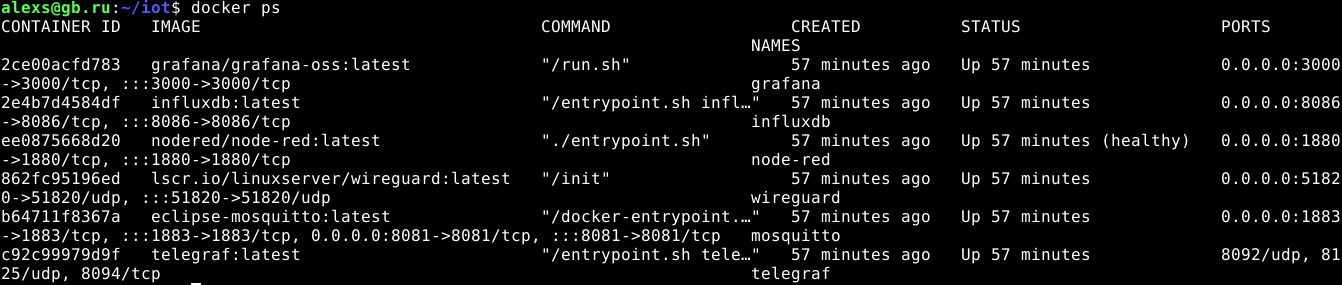
В соответствии с конфигурацией из *docker-compose.yml* файла создались сами контейнеры



Выполнив команду:

docker ps

Видим, что все контейнеры запущены успешно



# Визуализация

## Mosquitto

Проверка менеджера **mqtt**

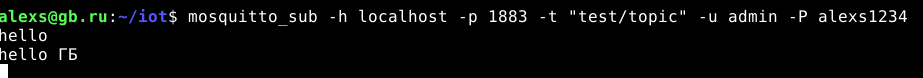
Откроем 2 терминала и в первом введем команду на создание потребителя запросов

mosquitto\_sub -h localhost -p 1883 -t "test/topic" -u admin -P alexs1234

а во втором другую команду на публикацию сообщений

mosquitto\_pub -h localhost -p 1883 -t "test/topic" -u admin -P alexs1234 -m "hello"

В первом терминале успешно будет принято сообщение



Москито работает хорошо.

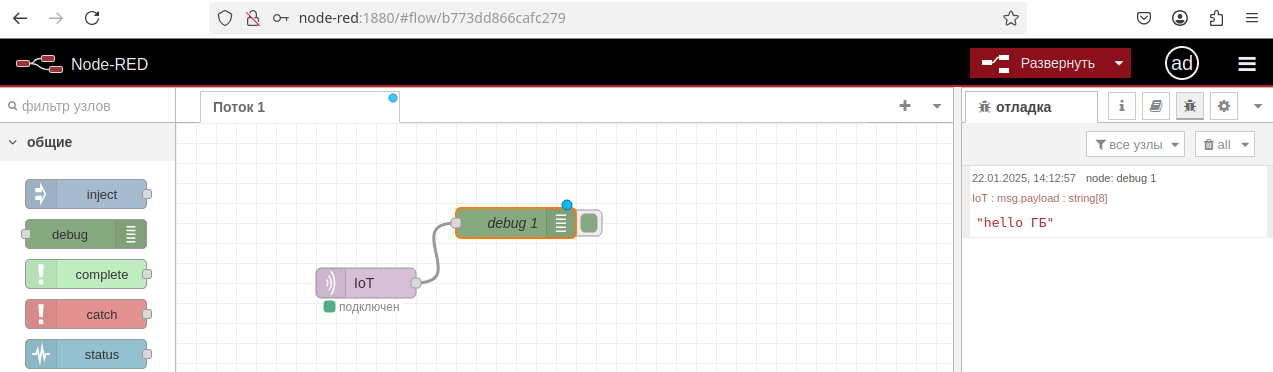
# Node-Red

## Прием сообщений

Используя логин **admin** и пароль **alexs1234** авторизуемся в **Node-Red** и добавим сетевой компонент **mqtt-in** в главный поток и подключим к нему компонент **dеbug** для вывода поступившей информации.

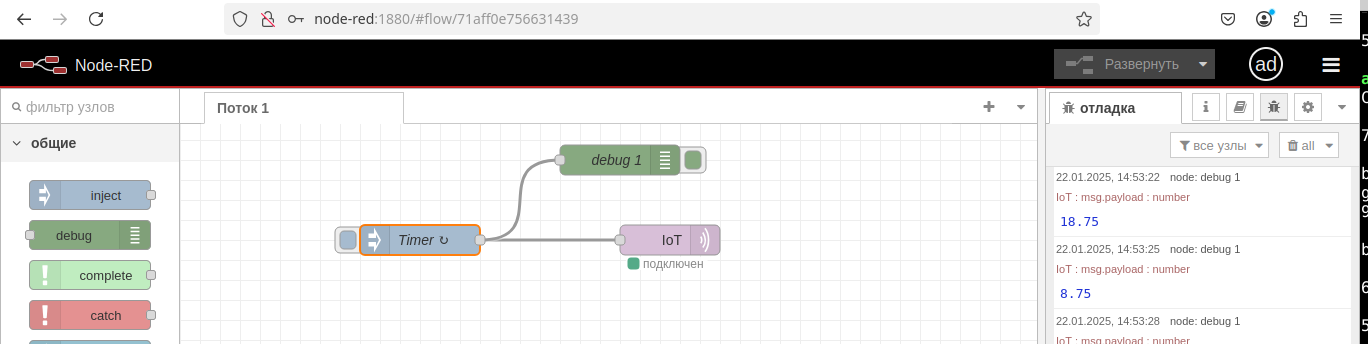
Параллельно в терминале введем команду

mosquitto\_pub -h localhost -p 1883 -t "IoT" -u admin -P alexs1234 -m "hello ГБ"

Информация успешно получена системой **Node-Red**

## Отправка сообщений

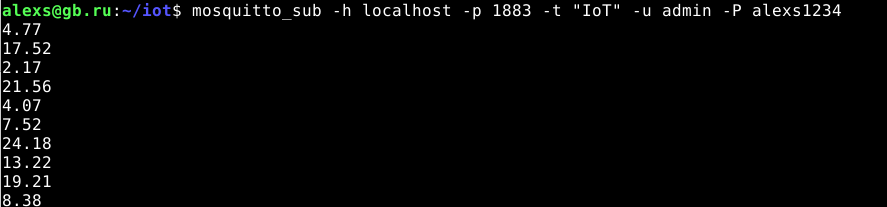
Изменим поток, зададим компонент временной метки, который генерирует рандомное число по формуле *$round(random()\*25,2)*. Получим рандомные значения похожие на температуру, перенаправим вывод значений в поток компонента **mqtt-out** и в **debug**



Выполнив команду в терминале

mosquitto\_sub -h localhost -p 1883 -t "IoT" -u admin -P alexs1234

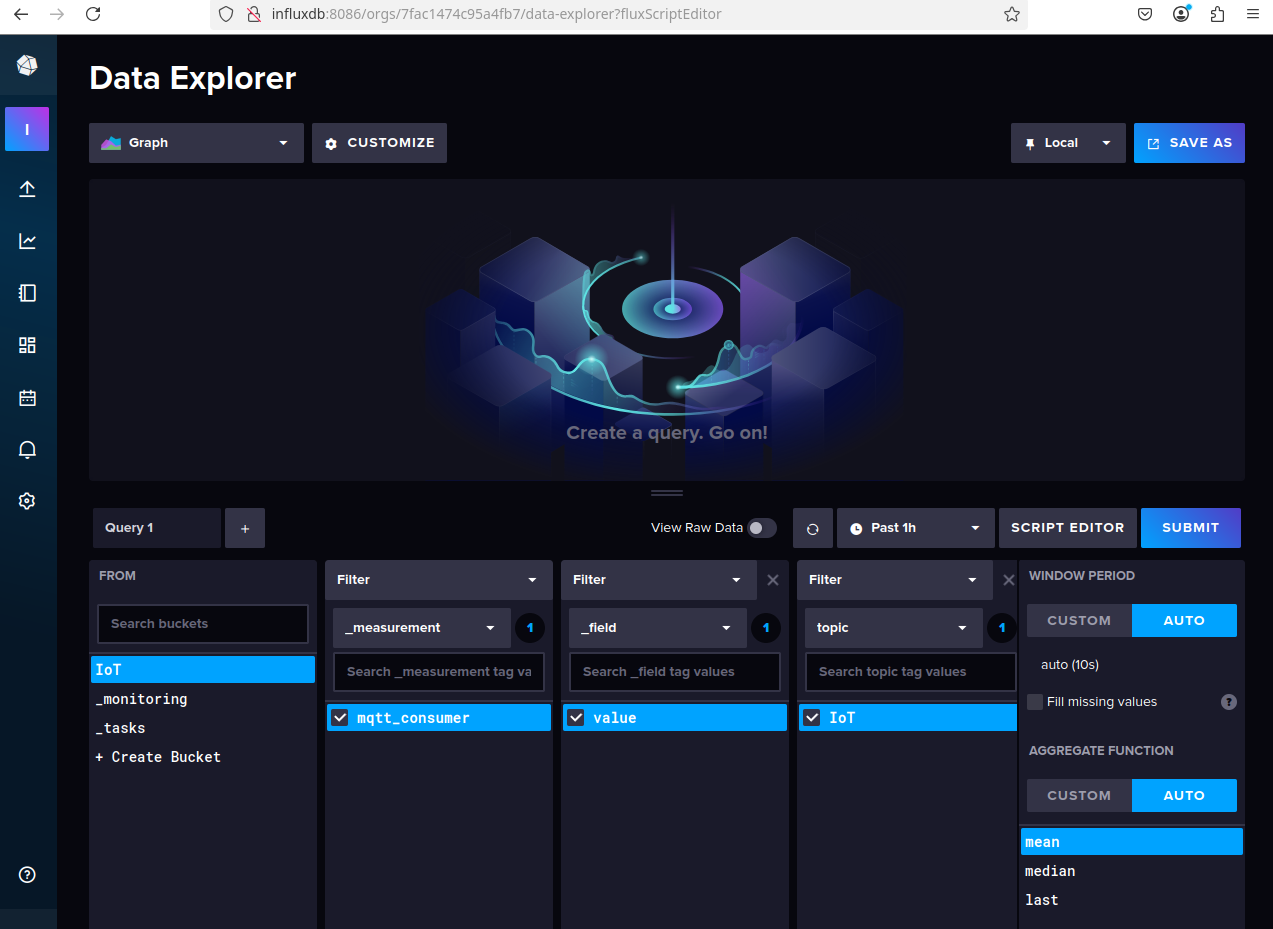
Получим вывод сгенерированных значений температуры



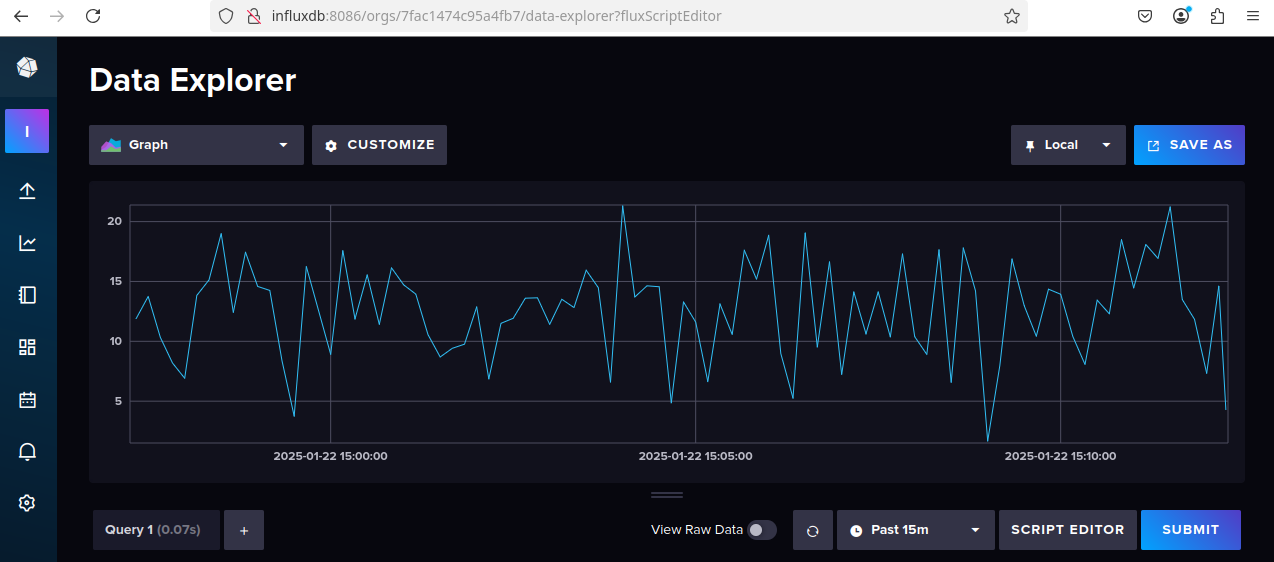
## Telegraf

Телеграф не имеет графического интерфейса. Он выступает в качестве подписчика на события mqtt-брокера и передает их в базу данных **InfluxDB2**.

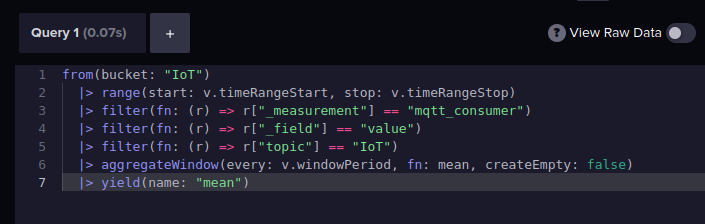
## InfluxDB2



Залогинившись в веб-интерфейс **InfluxDB** в разделе *Data Explorer* создадим запрос из выпадающих значений, после нажатия на кнопку **Submit** строится гистограмма полученных значений температуры от брокера



Нажав на кнопку **script editor**, перед нами предстает запрос на языке *Flux*



Если текст данного запроса скопировать, то в **Grafana** можно построить красивый график.

## Grafana

Войдя в графану нужно создать дашборт и в него добавить визуализацию. В качестве источника выбрать **InfluxDB** и добавить код ниже в раздел **Queries**

from(bucket: "IoT")

|> range(start: v.timeRangeStart, stop: v.timeRangeStop)

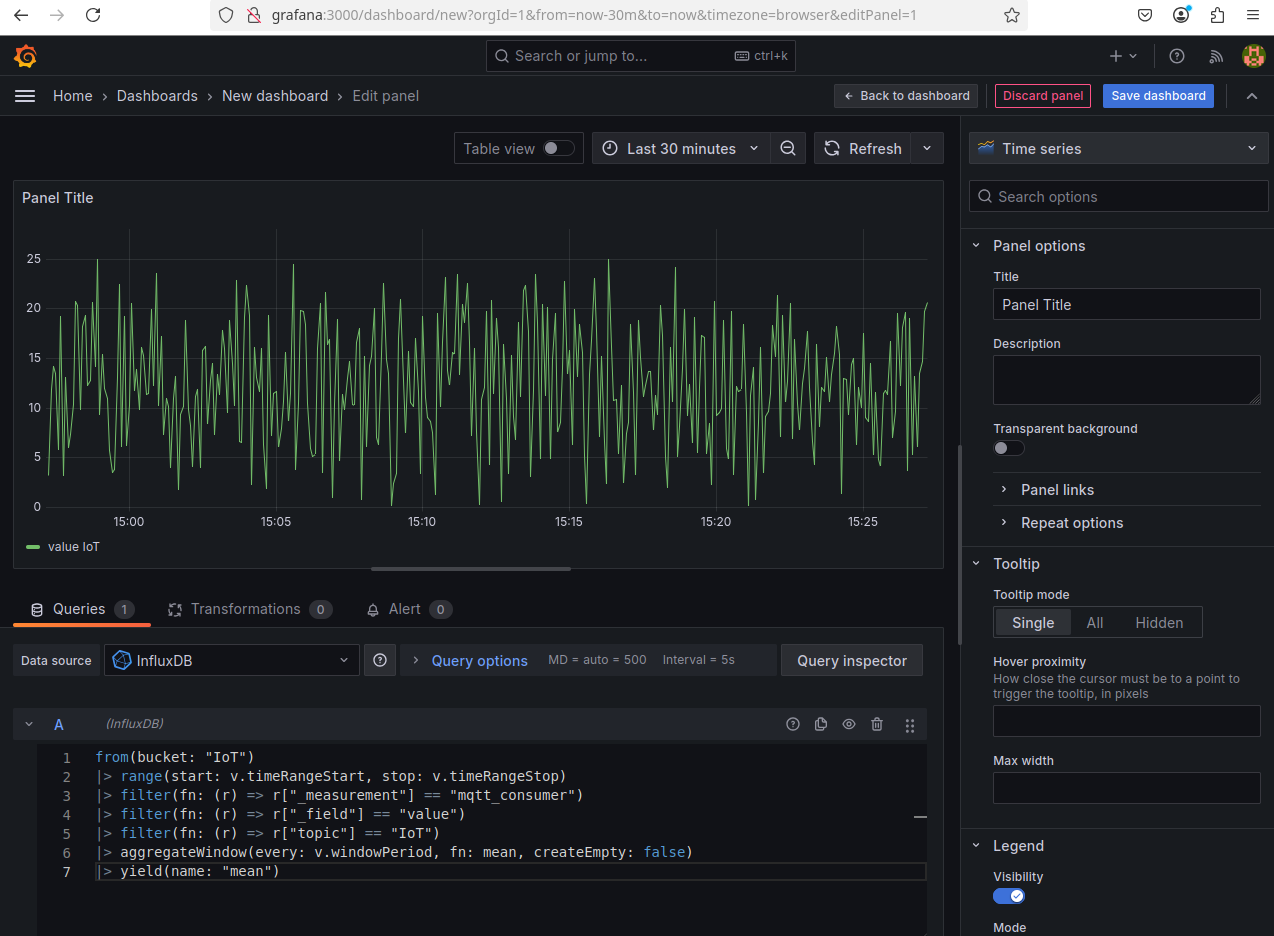
|> filter(fn: (r) => r["\_measurement"] == "mqtt\_consumer")

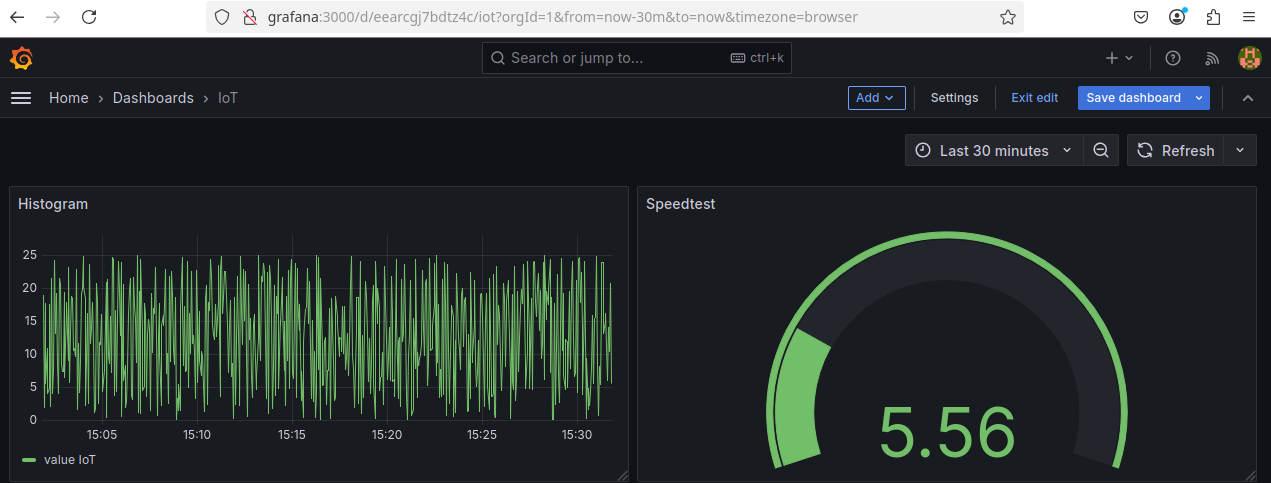
|> filter(fn: (r) => r["\_field"] == "value")

|> filter(fn: (r) => r["topic"] == "IoT")

|> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)

|> yield(name: "mean")

После этого можно сохранить визуализацию, и на основе того же запроса, создать спидометр

Таким образом, визуализировали показания сразу двумя способами от одной метрики.

В дашборде можно размещать большой количество информации о разных датчиках. Графана позволяет создавать огромное количество дашбордов и делится ими с друзьями.

В частности подключение можно сделать с помощью защищенного канала **WireGuard**. В моем случае отсутствует выделенный внешний IP-адрес поэтому доступ не предоставляю.

Но, **wireguard** все-равно был настроен вместе с остальными докер-контейнерами.

## WireGuard

Контейнер открывает случайны порт на компьютере и в случае наличия внешнего ip-адреса можно сделать редирект на данный порт и подключаться удаленно к системе визуализации.

В ранее созданной директории **wireguard** создает конфигурационные файлы, которые можно использовать для установки защищенного соединения. Создается сразу 5 конфигураций. Причем каждая имеет картинку с QR-кодом, который облегчает подключение.

