

# Разворачивание сетевого сервера + HomeAssistant + Grafana

## Установка и настройка БД, Node-red и брокеров

### 1) Установка СУБД, AMQP брокера, MQTT брокера и средства запуска Docker приложений:

```
apt install postgresql rabbitmq-server mosquitto docker-compose
```

### 2) Настройка PostgreSQL:

1. Открыть файл `/etc/postgresql/<version>/main/postgresql.conf` и установить значения следующих переменных:

```
listen_address = '*'  
max_connections = 500  
shared_buffers = 8192MB
```

`shared_buffers` нужно установить в 25-30% от объема доступной оперативной памяти. В данном случае общий объем 32G

2. Открыть файл `/etc/postgresql//main/pg_hba.conf` и добавить в конце строки:

```
host      all            all            172.16.0.0/12          scram-sha-256
```

При необходимости:

1. Заменить `password_encryption` на `scram-sha-256`
2. Заменить `md5` на `scram-sha-256`

3. Перезапустить PostgreSQL командой:

```
systemctl restart postgresql
```

4. Создать пользователя и базу данных, установить пароль пользователю и выдать права на созданную БД:

```
sudo su - postgres  
createuser airbit  
createdb airbit_server  
psql -c "alter user airbit with encrypted password '89hkhsdfH63d1q';"  
psql -c "alter schema public owner to airbit;"  
psql -c "alter database airbit_server owner to airbit;"
```

В версиях postgresql 15+ при необходимости выполнить:

```
psql -c "grant all on schema public to airbit;"  
psql -c "grant all privileges on database airbit_server to airbit;"
```

5. Создать расширения:

```
psql --dbname "airbit_server" -c "create extension ltree; create extension pgcrypto;"
```

6. Создаем пользователя и даем ему права для отдельной БД параметров, передаваемых в графану

```
sudo -u postgres createuser --login --pwprompt airpi  
sudo -u postgres psql -c "CREATE DATABASE parametrs WITH OWNER = airpi;"  
sudo -u postgres psql -c "GRANT ALL PRIVILEGES ON DATABASE parametrs TO airpi;"
```

Запросит пароль для пользователя. Пароль по умолчанию: 97HJhd1lsFF

### 3) Настройка RabbitMQ:

1. Включить Management Plugin для доступа в веб-интерфейс:

```
rabbitmq-plugins enable rabbitmq_management
```

2. Создать пользователя и выдать права на виртуальный хост "/":

```
rabbitmqctl add_user airbit 97HJhd1sFF
rabbitmqctl set_user_tags airbit administrator
rabbitmqctl set_permissions -p / airbit ".*" ".*" ".*"
```

3. Удалить гостевую учетную запись:

```
rabbitmqctl delete_user guest
```

#### 4) Настройка MQTT:

1. В файл */etc/mosquitto/mosquitto.conf* прописать:

```
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example
pid_file /var/run/mosquitto/mosquitto.pid
persistence true
persistence_location /var/lib/mosquitto/
log_dest file /var/log/mosquitto/mosquitto.log
log_type error
log_type warning
log_type notice
log_type information
log_type websockets
websockets_log_level 0
include_dir /etc/mosquitto/conf.d
```

2. Создать файл командой:

```
touch /etc/mosquitto/conf.d/default.conf
```

В файл поместить:

```
listener 1883
```

3. Создать файл командой:

```
touch /etc/mosquitto/conf.d/password_auth.conf
```

В файл поместить:

```
allow_anonymous false
password_file /etc/mosquitto/passwd
```

Пункты 4-5 опционально, только если требуется SSL и имеется сертификат

4. В файл */etc/mosquitto/conf.d/ssl.conf* прописать:

```
listener 8883
cafile /etc/ssl/certs/ca.file
certfile /etc/ssl/certs/public_cert.crt
keyfile /etc/ssl/certs/private_key.key
```

5. В файл */etc/mosquitto/conf.d/websocket.conf* прописать:

```
listener 9001
protocol websockets
http_dir /var/lib/mosquitto/www

listener 9883
```

```
protocol websockets
http_dir /var/lib/mosquitto/www
cafile /etc/ssl/certs/ca.file
certfile /etc/ssl/certs/public_cert.crt
keyfile /etc/ssl/certs/private_key.key
```

## 6. Создать пользователя MQTT:

Пароль будет запрошен, устанавливаем как для RabbitMQ

```
mosquitto_passwd -c /etc/mosquitto/passwd airbit
chmod 0640 /etc/mosquitto/passwd
chown root:mosquitto /etc/mosquitto/passwd
```

## 7. Перезапустить mosquitto:

```
systemctl restart mosquitto
```

## 5) Настройка Node-RED:

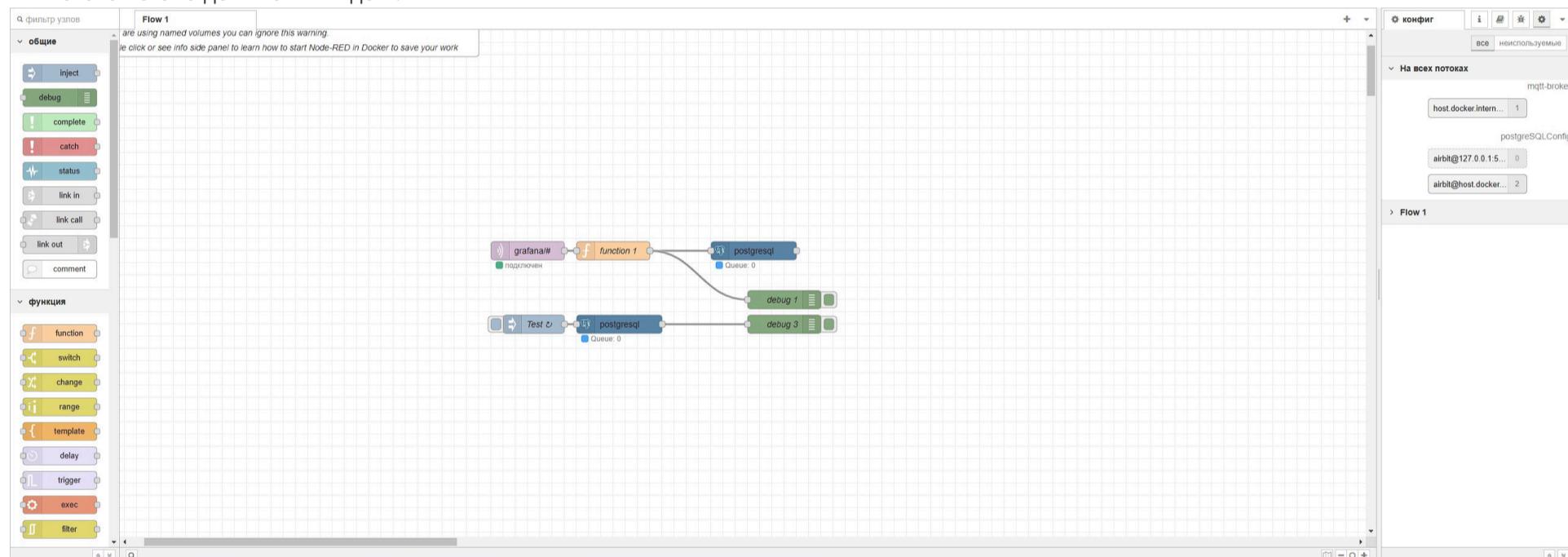
### 1. Установить Node-RED:

```
docker run -d -p 1880:1880 --add-host=host.docker.internal:host-gateway --restart=unless-stopped --name mynodered nodered/node-red
```

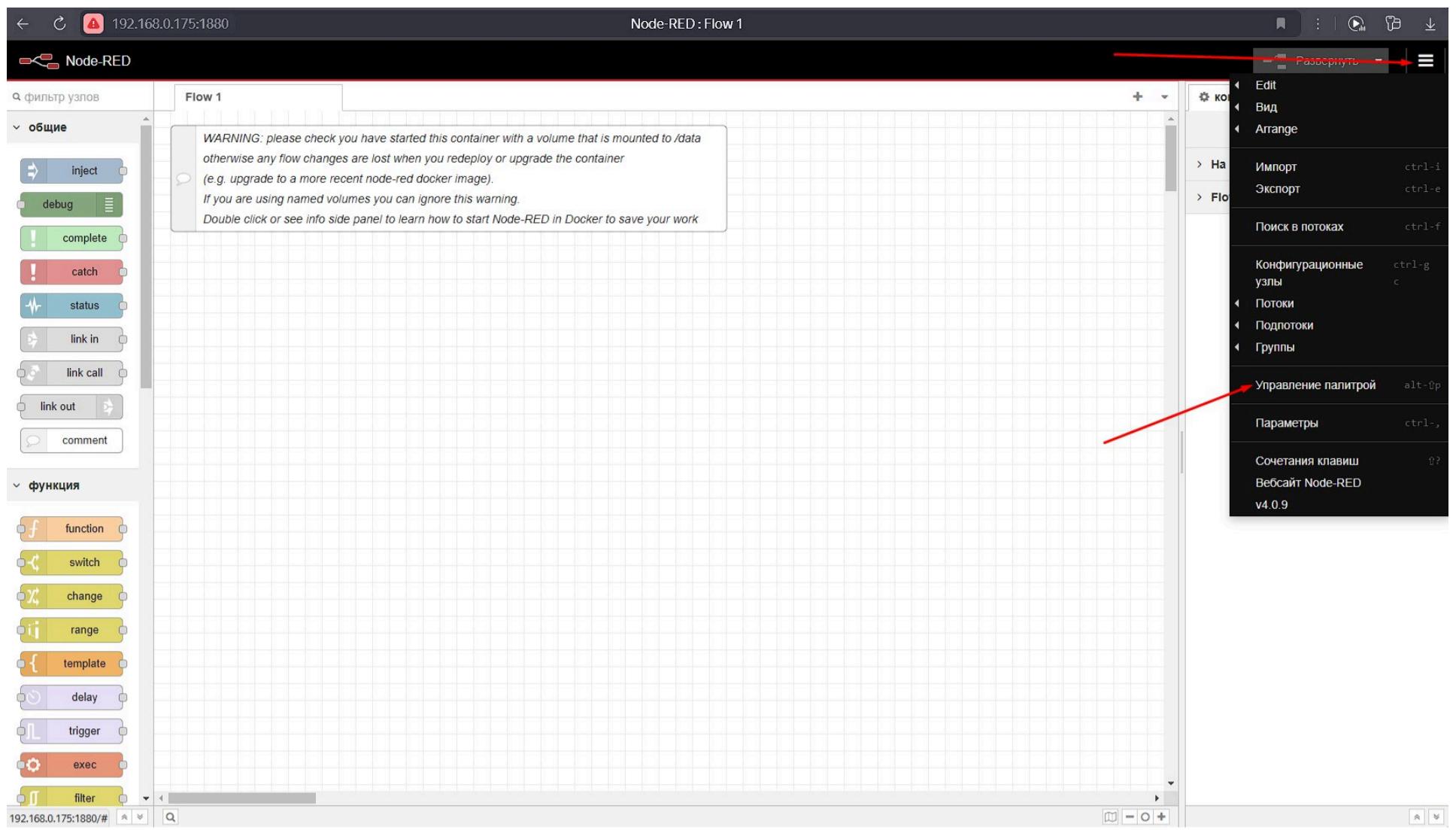
### 2. Зайти в Node-RED: <ip адрес локальной установки>:1880

### 3. Настраиваем логику согласно инструкции:

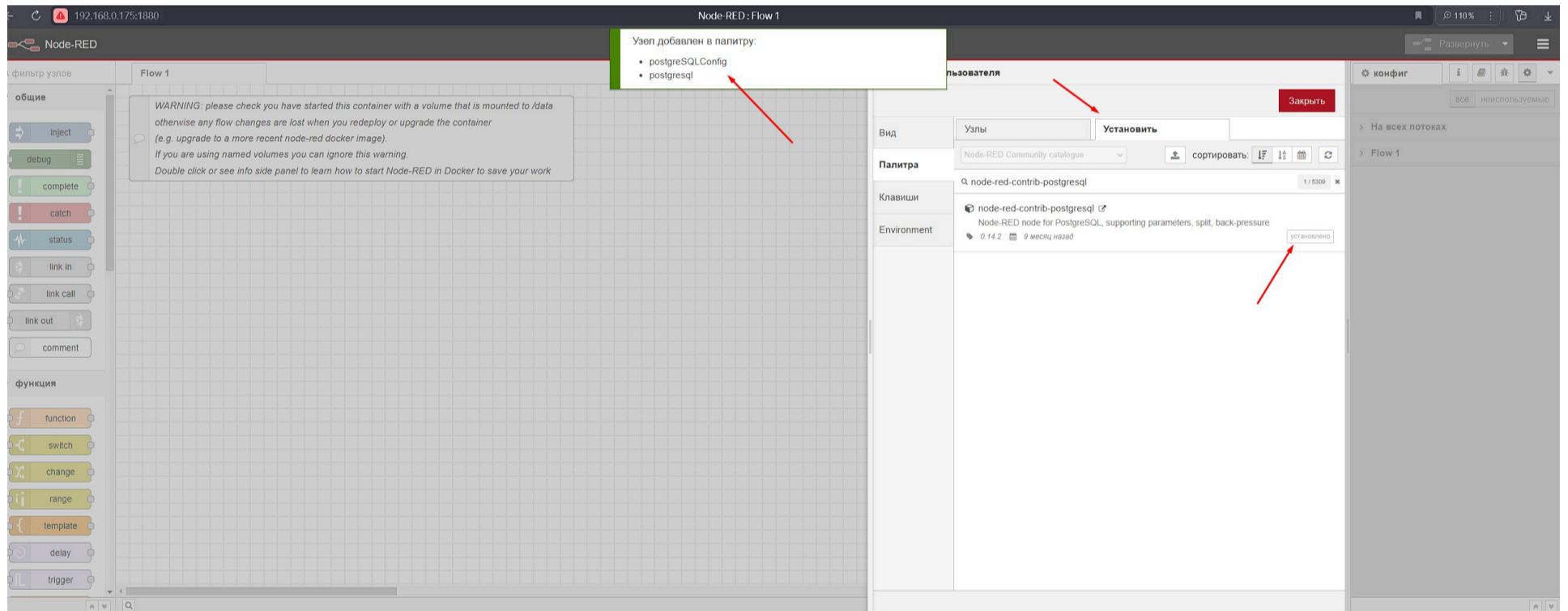
#### ▼ Итоговая схема должна выглядеть:



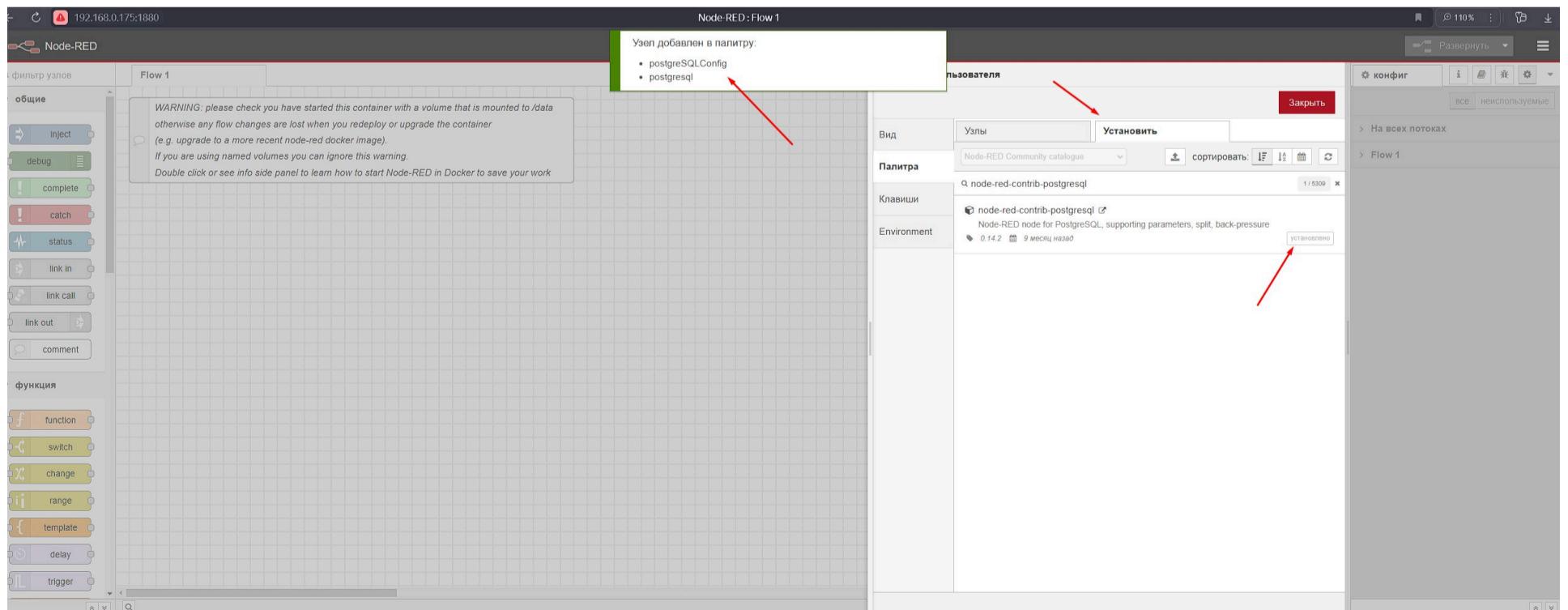
#### ▼ Шаг 1



### ▼ Шаг 2

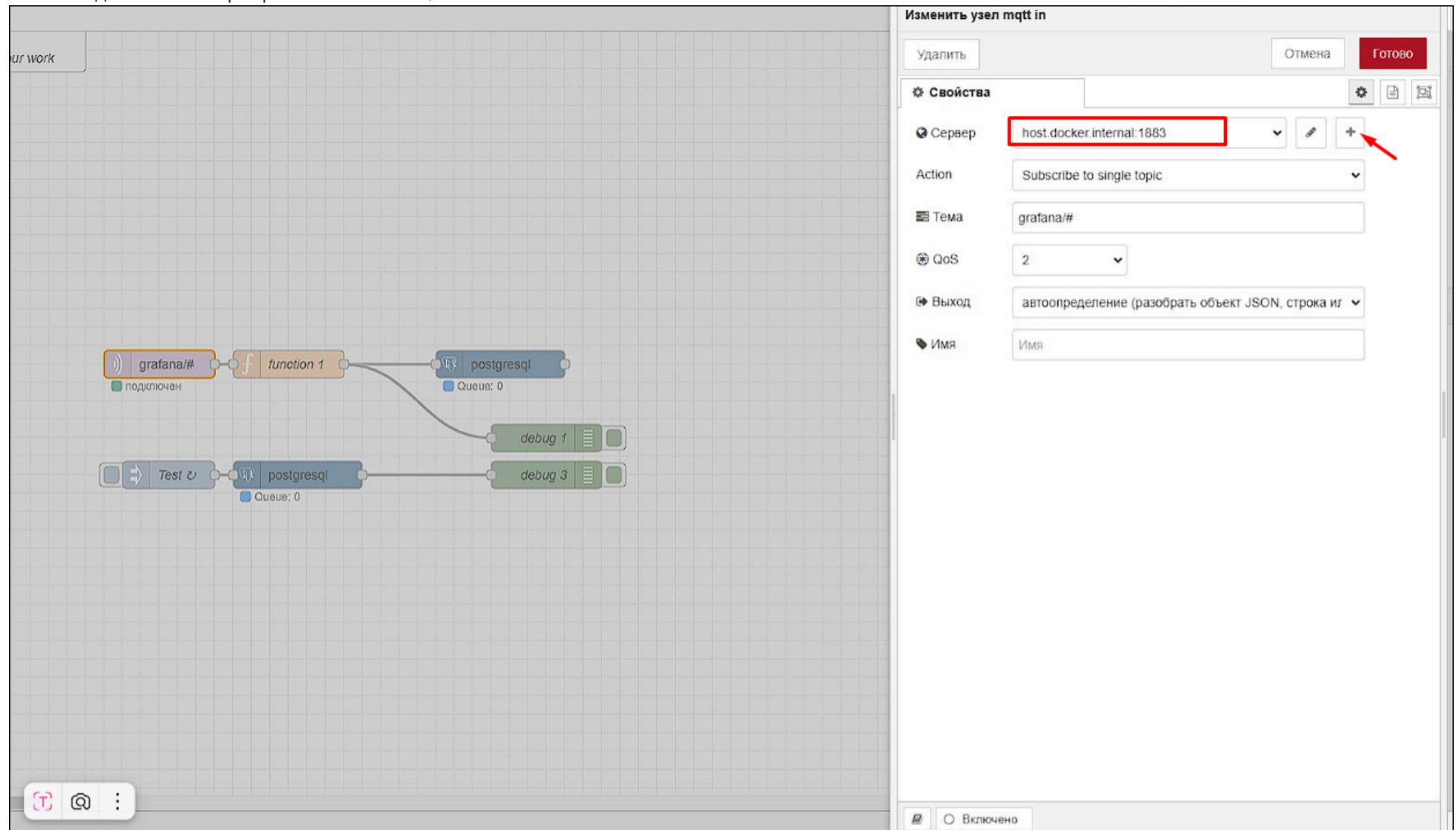


### ▼ Шаг 3



### ▼ Шаг 4

Сначала добавляем сервер и пользователя, затем заполняем поля



Изменить узел mqtt in > Изменить узел mqtt-broker

Удалить	Отмена	Обновить
<b>Свойства</b>		
<b>Имя</b>	Имя	
Соединение	Безопасность	Сообщения
Имя польз.	airbit	
Пароль	.....	

Инфо

Потоки  
Подпотоки  
Глобальные конфиг узлы  
mqtt-broker  
postgreSQLConfig

host.docker.internal:1883

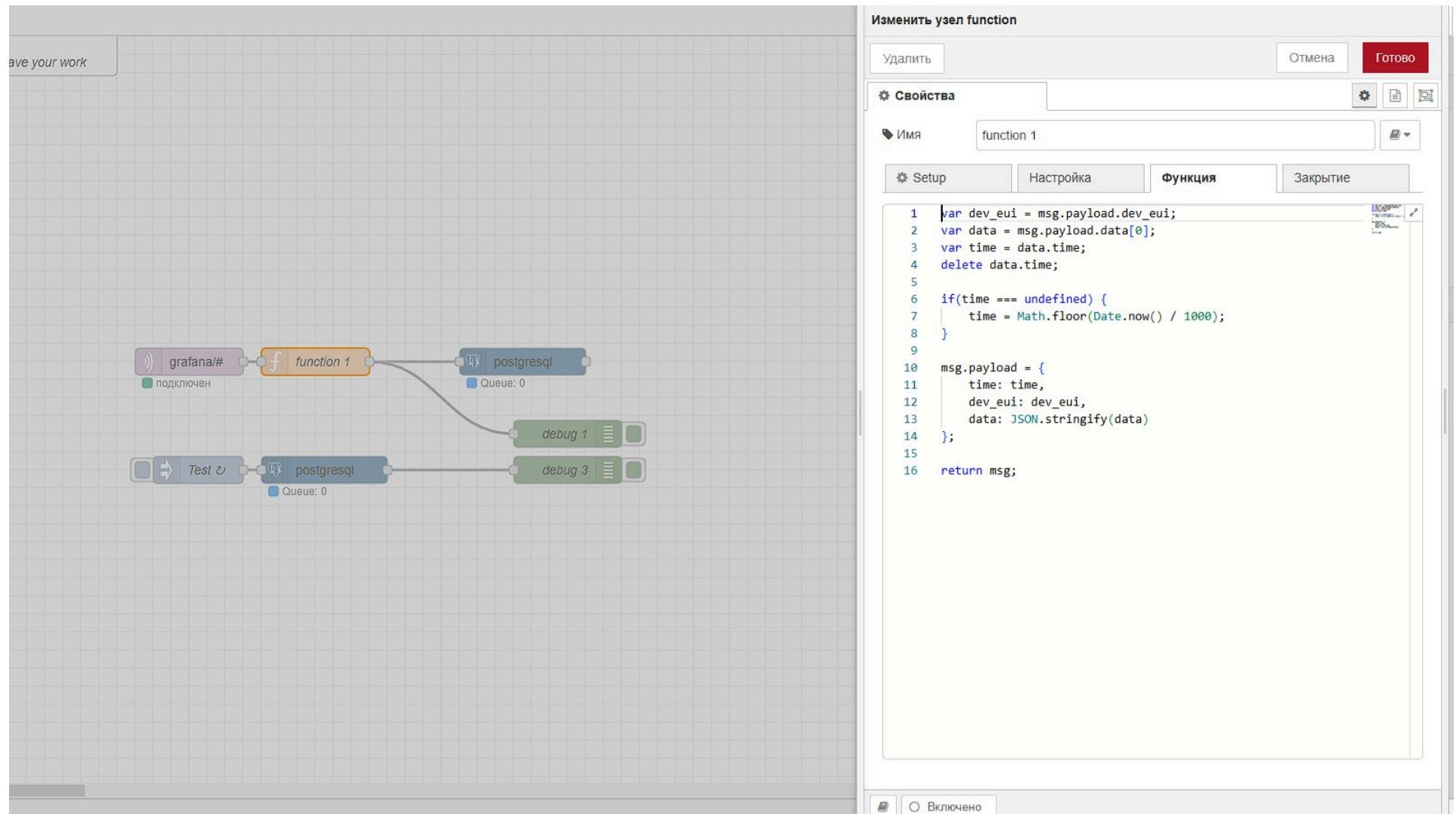
Узел	"9af11287ced1c8c5"
Тип	mqtt-broker

показать больше

Импортируйте поток, перетаскивая его JSON в редактор или с помощью **ctrl-i**

Логин и пароль от пользователя MQTT брокера

## ▼ Шаг 5



В функцию шаг 5 копируем:

```
var dev_eui = msg.payload.dev_eui;
var data = msg.payload.data[0];
var time = data.time;
delete data.time;

if(time === undefined) {
    time = Math.floor(Date.now() / 1000);
}

msg.payload = {
    time,
    dev_eui: dev_eui,
    data: JSON.stringify(data)
};

return msg;
```

## ▼ Шаг 6

Сначала добавляем сервер и пользователя, затем заполняем поля

The screenshot shows the Airbitz configuration interface. On the left, a flowchart is displayed with nodes: 'grafana/#' (purple), 'function 1' (orange), 'postgresql' (blue), 'Test' (grey), and 'debug 1' (green). Arrows indicate data flow from 'grafana/#' through 'function 1' to 'postgresql', and from 'Test' to 'debug 1'. The 'postgresql' node has a status message 'Queue: 0'.

On the right, a modal dialog titled 'Изменить узел postgresql' (Change node PostgreSQL) is open. It contains fields for 'Name' (Имя), 'Server' (airbit@host.docker.internal:5432), and 'Query'. The 'Query' field contains the following SQL code:

```
1 INSERT INTO sensor_data (dev_eui, data, time)
2 VALUES ('{{msg.payload.dev_eui}}', '{{msg.payload.data}}')
```

A red arrow points to the '+' button in the top right corner of the dialog window.

Изменить узел postgresql > Изменить узел PostgreSQLConfig

Удалить

Отмена

Обновить

Свойства



Name

dbConnection

Connection

Security

Pool

Host

host.docker.internal

Port

5432

Database

parametrs

SSL

false

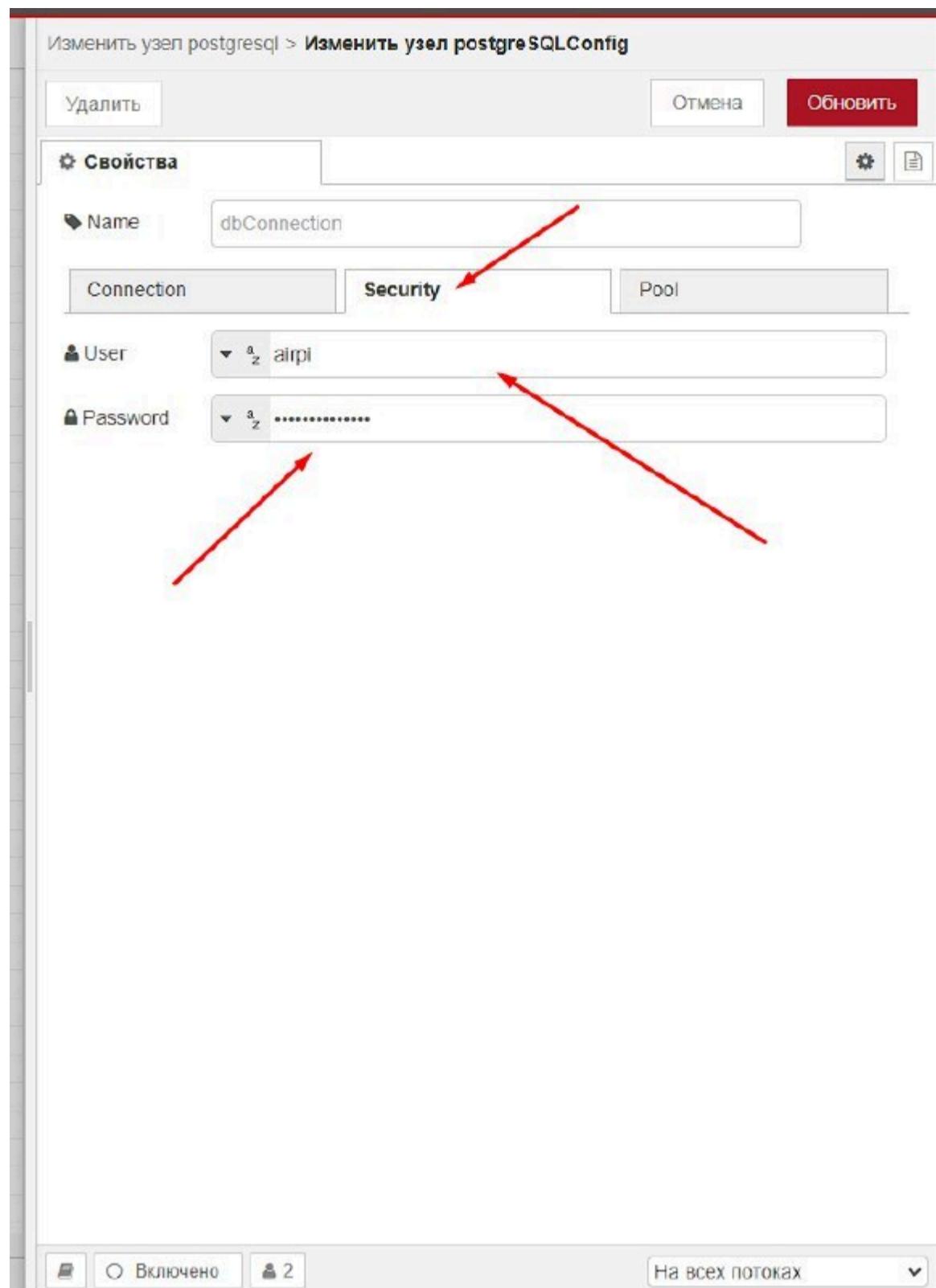


Включен

2

На всех потоках



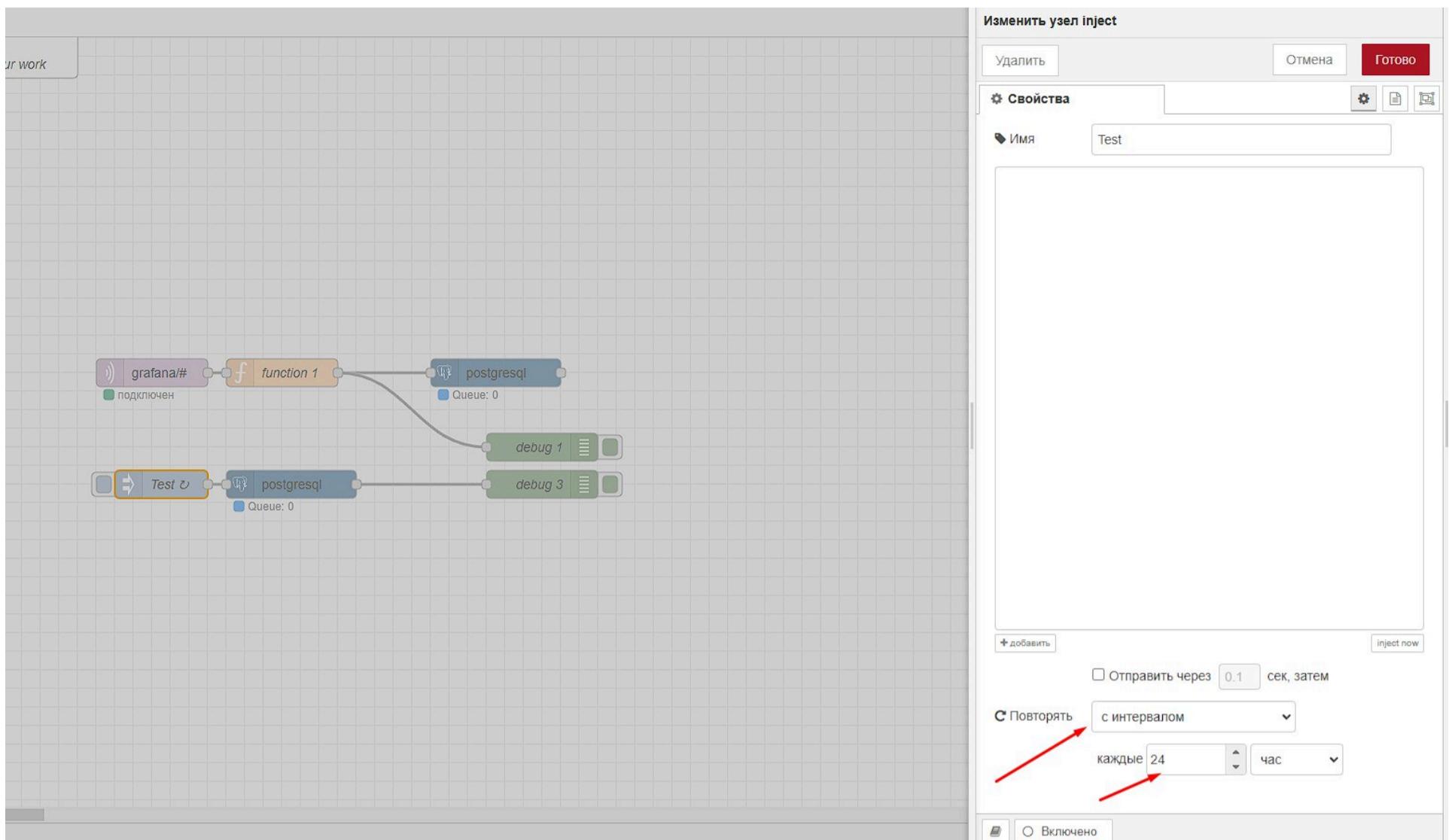


Логин и пароль от пользователя БД с названием "parametrs"

В функцию шаг 6 копируем:

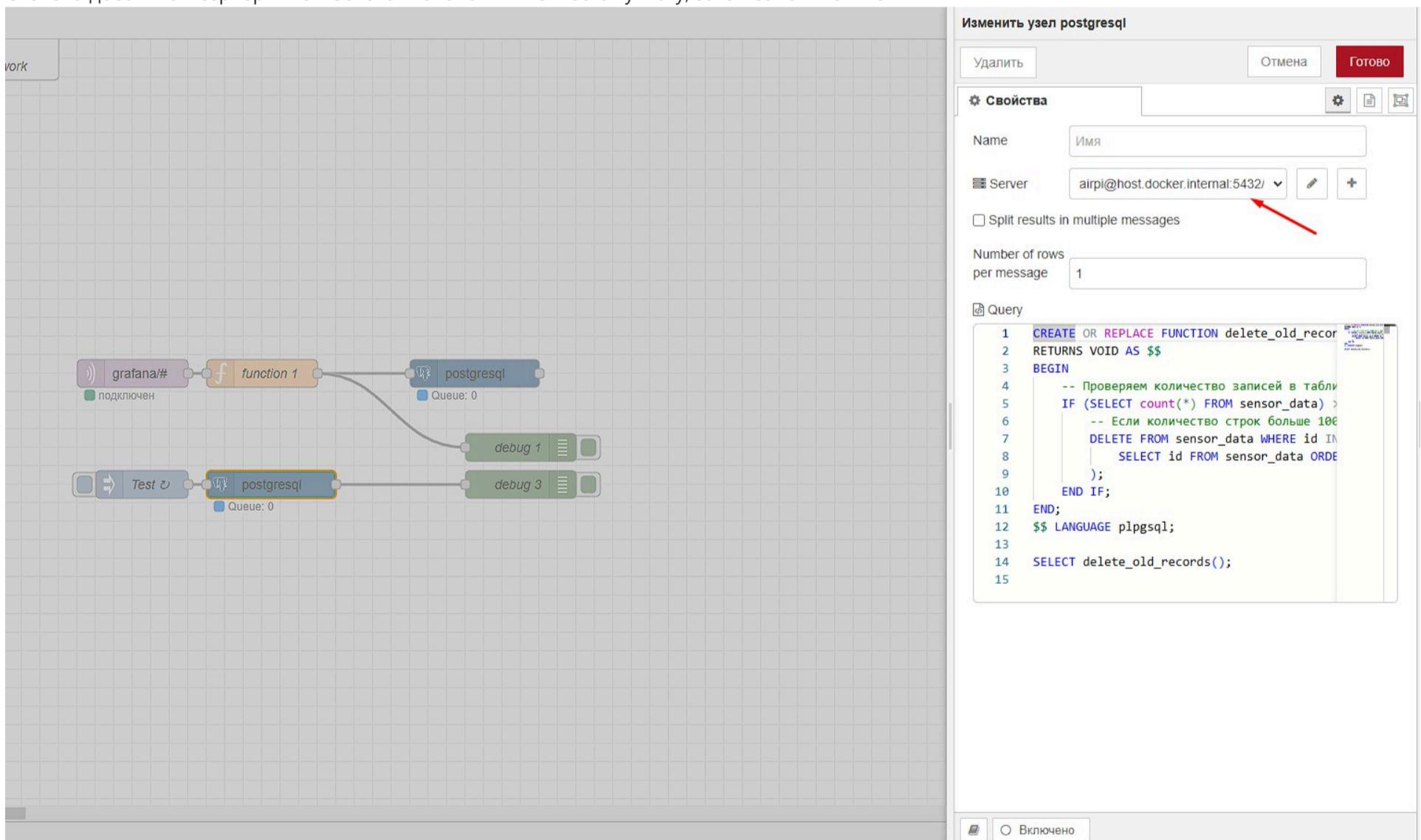
```
INSERT INTO sensor_data (dev_eui, data, times)
VALUES ('{{msg.payload.dev_eui}}', ('{{{msg.payload.data}}})::json), to_timestamp('{{msg.payload.time}}');
```

▼ Шаг 7



#### ▼ Шаг 8

Сначала добавляем сервер и пользователя аналогичные шестому шагу, затем заполняем поля



В функцию шаг 8 копируем:

```

CREATE OR REPLACE FUNCTION delete_old_records()
RETURNS VOID AS $$$
BEGIN
    -- Проверяем количество записей в таблице
    IF (SELECT count(*) FROM sensor_data) > 1000000 THEN
        -- Если количество строк больше 1000, удаляем 100 самых старых записей
        DELETE FROM sensor_data WHERE id IN (
            SELECT id FROM sensor_data ORDER BY times ASC LIMIT ((SELECT count(*) FROM sensor_data)-1000000)
        );
    END IF;
END;
$$ LANGUAGE plpgsql;

```

```
SELECT delete_old_records();
```

## Установка сетевого сервера

1. Командой создать папку в директории пользователя:

```
mkdir ~/lns  
cd ~/lns
```

2. Скопировать в каталоги файлы docker-compose.yaml и .env . Актуальные версии файлов лежат по адресу: <https://git.air-bit.eu/airbit/lora-server/-/tree/develop/contrib/docker/prod>

3. Сконфигурировать .env файл:

- Указать Version (Текущую актуальную версию)
- FRONTEND\_HOSTNAME (lns-pi.air-bit.local)
- Имена хостов сменить на host.docker.internal
- Пароли выставить аналогичные заданным для БД, брокеров
- Установить логин и пароль для #HTTP REST API (Пароль: Fyh!236G)
- Сменить названия БД
- Настроить HTTP\_DOWNLINK\_URI=<http://airpi@air-bit.eu:airpi62@host.docker.internal/api/data/>

При необходимости в docker-compose yaml проверить в environment контейнеров, которые используют redis наличие CACHE\_URL (CACHE\_URL=<redis://redis:6379/0>)

4. Создать папку для SSL сертификатов:grafa

```
mkdir -p /etc/ssl/lns-pi.air-bit.local
```

5. Создать само подписные сертификаты в созданной папке:

```
openssl req -x509 -newkey rsa:4096 -keyout private.key -out public.crt -days 3650 -nodes -subj "/CN=lns-pi.air-bit.local"
```

6. Авторизоваться командой:

Будет запрошен логин и пароль на наш gitlab

```
docker login registry.air-bit.eu/v2/airbit/lora-server
```

7. Проверочно запустить LNS командой:

```
docker-compose up
```

Если запуск произошел с ошибкой, выполнить пункт 8

8. Посмотреть логи командой:

```
docker-compose logs
```

или

```
docker-compose logs <name-of-service>
```

Где *name-of-service* имя сервиса который запустился с ошибкой

9. Создать суперадмина:

```
docker run -it \  
-e DATABASE_URI=postgresql://airbit:89hkhdfH63d1q@172.17.0.1:5432/airbit_server \  
registry.air-bit.eu/airbit/lora-server/lns-base:latest python manage.py create_admin -i
```

1. При необходимости latest заменить на устанавливаемую версию
2. Стандартное логин: [airpi@air-bit.ru](mailto:airpi@air-bit.ru) пароль: airpi62

10. Добавить периодические задания в cron: Под пользователем root открыть файл командой:

```
sudo crontab -e
```

Внести в файл:

```
0    3    *    *    *    docker run --add-host=host.docker.internal:host-gateway --env-file ~/lns/.env registry.air-bit.eu/ai
0    3    1    *    *    docker run --add-host=host.docker.internal:host-gateway --env-file ~/lns/.env registry.air-bit.eu/ai
0    6    1    *    *    docker run --add-host=host.docker.internal:host-gateway --env-file ~/lns/.env registry.air-bit.eu/ai
5    *    *    *    *    docker run --add-host=host.docker.internal:host-gateway --env-file ~/lns/.env registry.air-bit.eu/ai
5    *    *    *    *    docker run --add-host=host.docker.internal:host-gateway --env-file ~/lns/.env registry.air-bit.eu/ai
5    *    *    *    *    docker run --add-host=host.docker.internal:host-gateway --env-file ~/lns/.env registry.air-bit.eu/ai
5    *    *    *    *    docker run --add-host=host.docker.internal:host-gateway --env-file ~/lns/.env registry.air-bit.eu/ai
5    *    *    *    *    docker run --add-host=host.docker.internal:host-gateway --env-file ~/lns/.env registry.air-bit.eu/ai
5    *    *    *    *    docker run --add-host=host.docker.internal:host-gateway --env-file ~/lns/.env registry.air-bit.eu/ai
5    *    *    *    *    docker run --add-host=host.docker.internal:host-gateway --env-file ~/lns/.env registry.air-bit.eu/ai
5    *    *    *    *    docker run --add-host=host.docker.internal:host-gateway --env-file ~/lns/.env registry.air-bit.eu/ai
5    3    *    *    *    docker run --add-host=host.docker.internal:host-gateway --env-file ~/lns/.env registry.air-bit.eu/ai
5    3    *    *    *    docker run --add-host=host.docker.internal:host-gateway --env-file ~/lns/.env registry.air-bit.eu/ai
5    3    *    *    *    docker run --add-host=host.docker.internal:host-gateway --env-file ~/lns/.env registry.air-bit.eu/ai
```

При необходимости в lns-base:latest latest заменить на развернутую версию

11. После всех проверок для запуска в режиме демона, запустить командой:

```
docker-compose up -d
```

12. Выйти из учетной записи gitlab после окончания работ:

```
docker logout
```

## Установка и настройка Grafana и HomeAssistant

### 1) Установка и настройка Grafana

1. Установить Grafana командой:

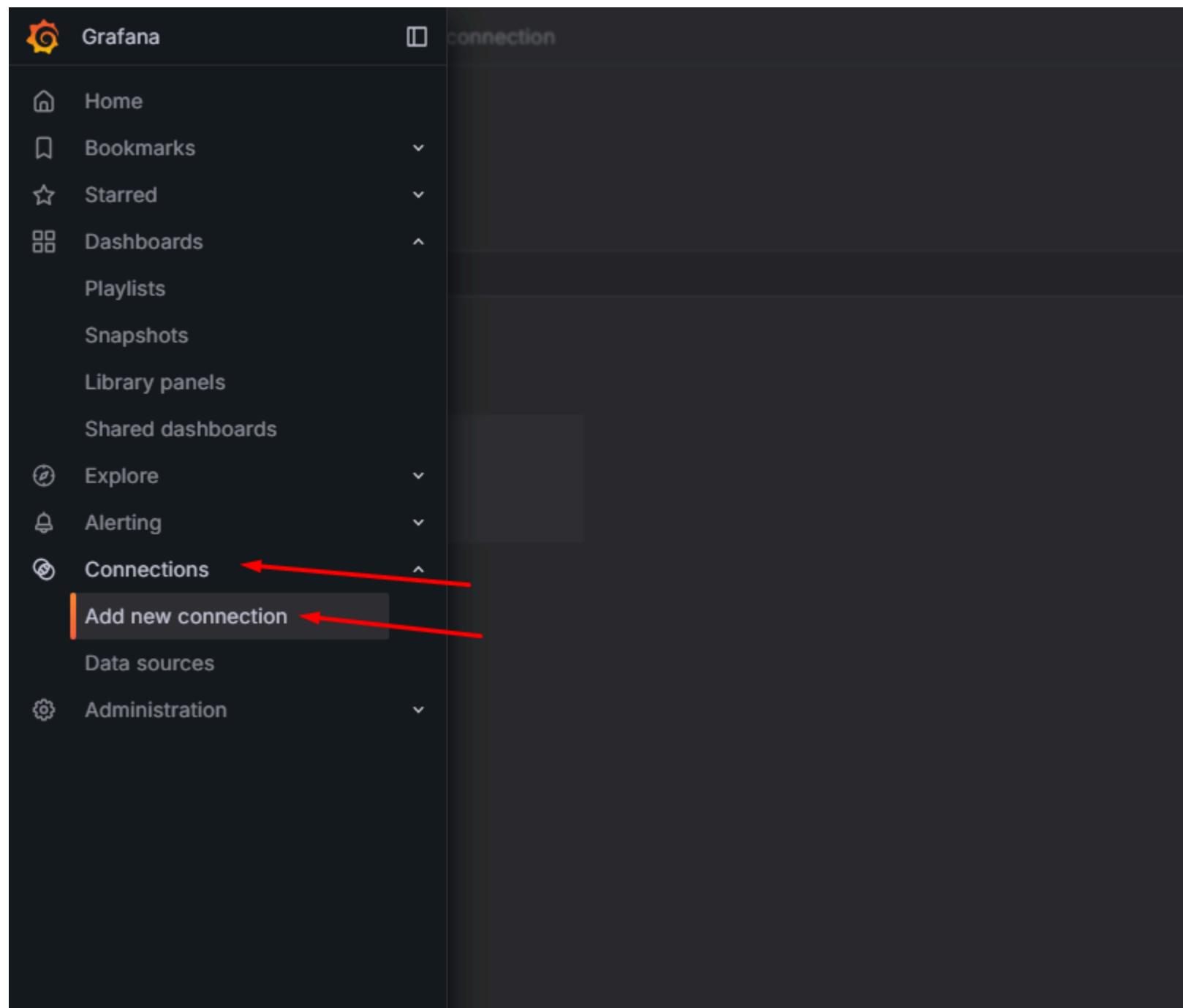
```
docker run -d \
--restart=unless-stopped \
-p 3000:3000 \
--name=grafana \
-e GF_SECURITY_ADMIN_USER=airpi \
-e GF_SECURITY_ADMIN_PASSWORD=airpi62 \
grafana/grafana
```

2. Зайти и авторизоваться в Grafana: <ip адрес локальной установки>:3000

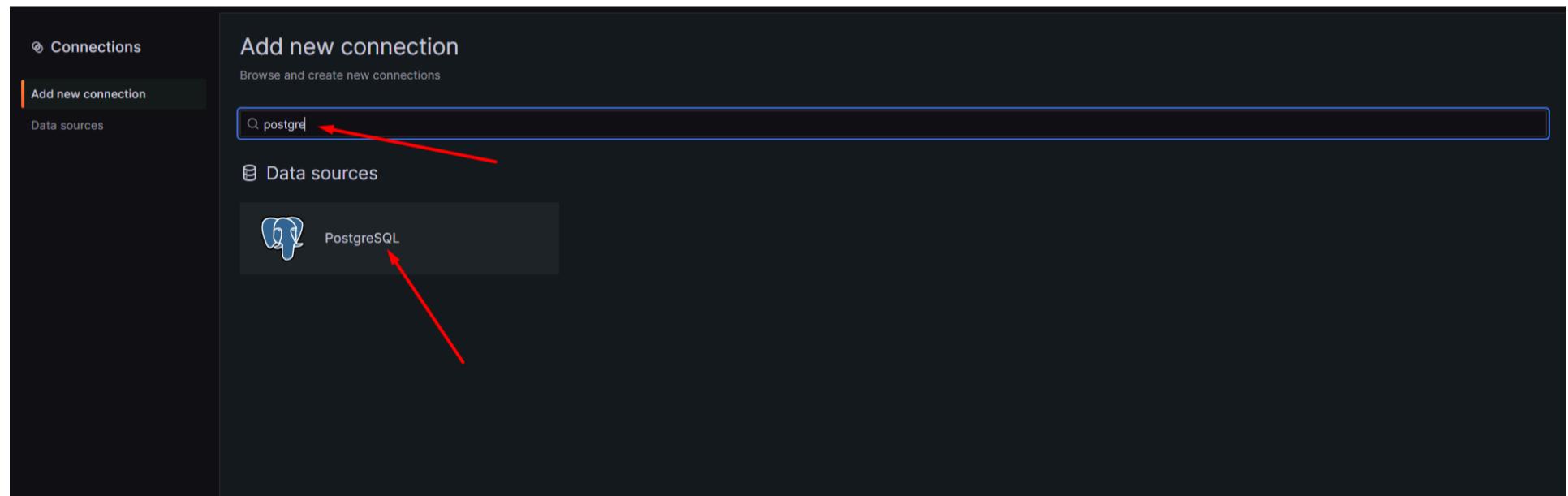
По умолчанию логин:airpi и пароль: 97HJhd1lsFF

3. Добавление поддержки PostgreSQL

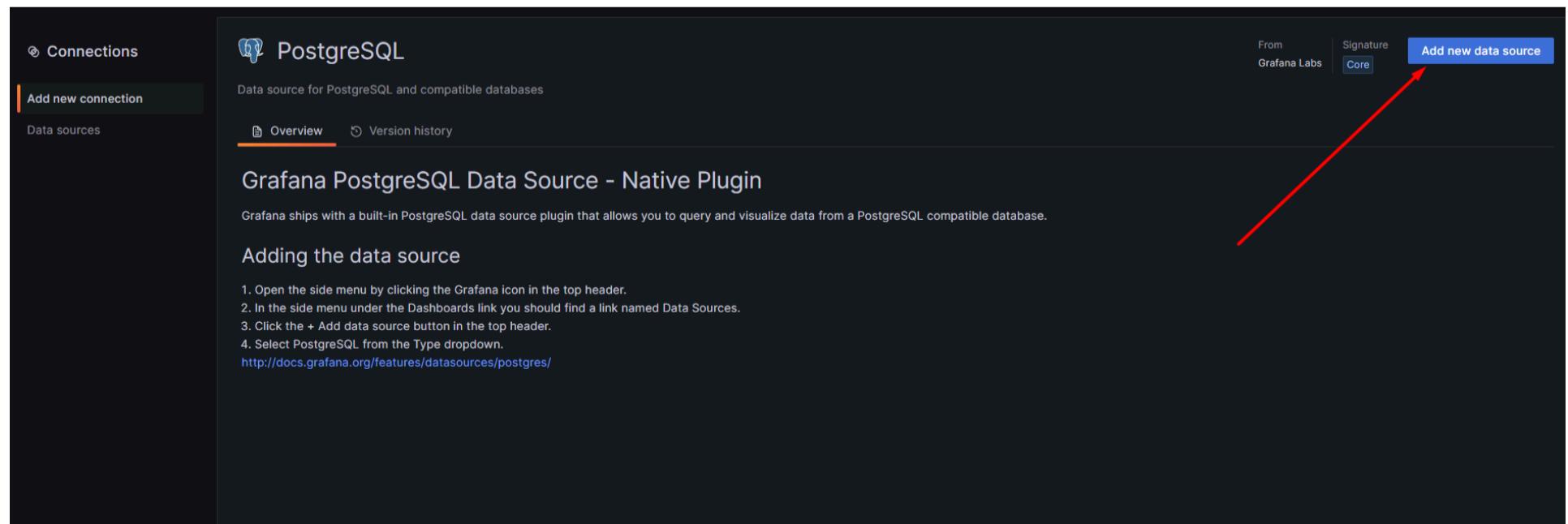
▼ Шаг 1



▼ Шаг 2

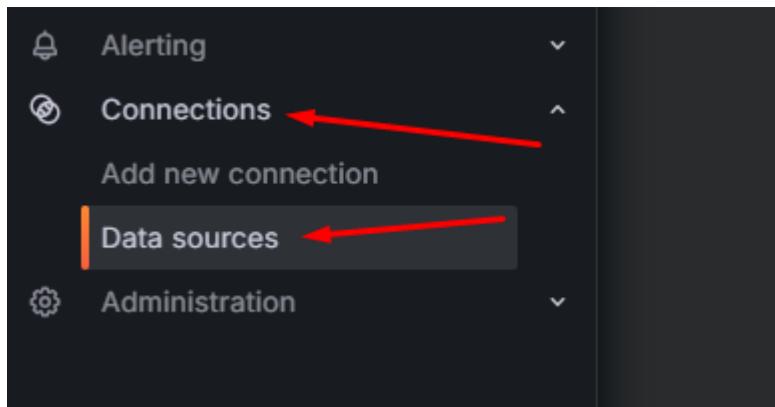


▼ Шаг 3



4. Добавить новый источник

▼ Шаг 1



▼ Шаг 2

This screenshot shows the 'Data sources' page. At the top right is a search bar with 'ctrl+k' and a 'Clear' button. Below it is a 'Sort by A-Z' dropdown. A red arrow points to the '+ Add new data source' button in the top right corner. The main area displays a yellow cartoon character icon with a magnifying glass over a database schema, and the text 'No data sources found'.

▼ Шаг 3

This screenshot shows the 'Add data source' dialog. At the top is a search bar with 'Q post' and a 'Clear' button. A red arrow points to the search bar. Below the search bar is a list of data source types. The first item is 'PostgreSQL' with its icon (blue elephant logo) highlighted with a red arrow. The text next to it reads 'Data source for PostgreSQL and compatible databases' and 'Core'.

▼ Шаг 4

Название БД с параметрами: "parametrs"

По умолчанию логин: airpi и пароль: 97HJhd1lsFF

The screenshot shows the 'grafana-postgresql-datasource' configuration page. At the top, it indicates the Type is PostgreSQL and Alerting is Supported. Below this, the 'Settings' tab is selected. The 'Name' field is set to 'grafana-postgresql-datasource'. A note at the top states: 'Before you can use the Postgres data source, you must configure it below or in the config file. For detailed instructions, [view the documentation](#)'. It also notes that fields marked with \* are required.

**User Permissions**: A note says: 'The database user should only be granted SELECT permissions on the specified database & tables you want to query. Grafana does not validate that queries are safe so queries can contain any SQL statement. For example, statements like `DELETE FROM user;` and `DROP TABLE user;` would be executed. To protect against this we Highly recommend you create a specific PostgreSQL user with restricted permissions. Check out the docs for more information.'

**Connection**: Host URL is 'host.docker.internal', Database name is 'parametrs'.

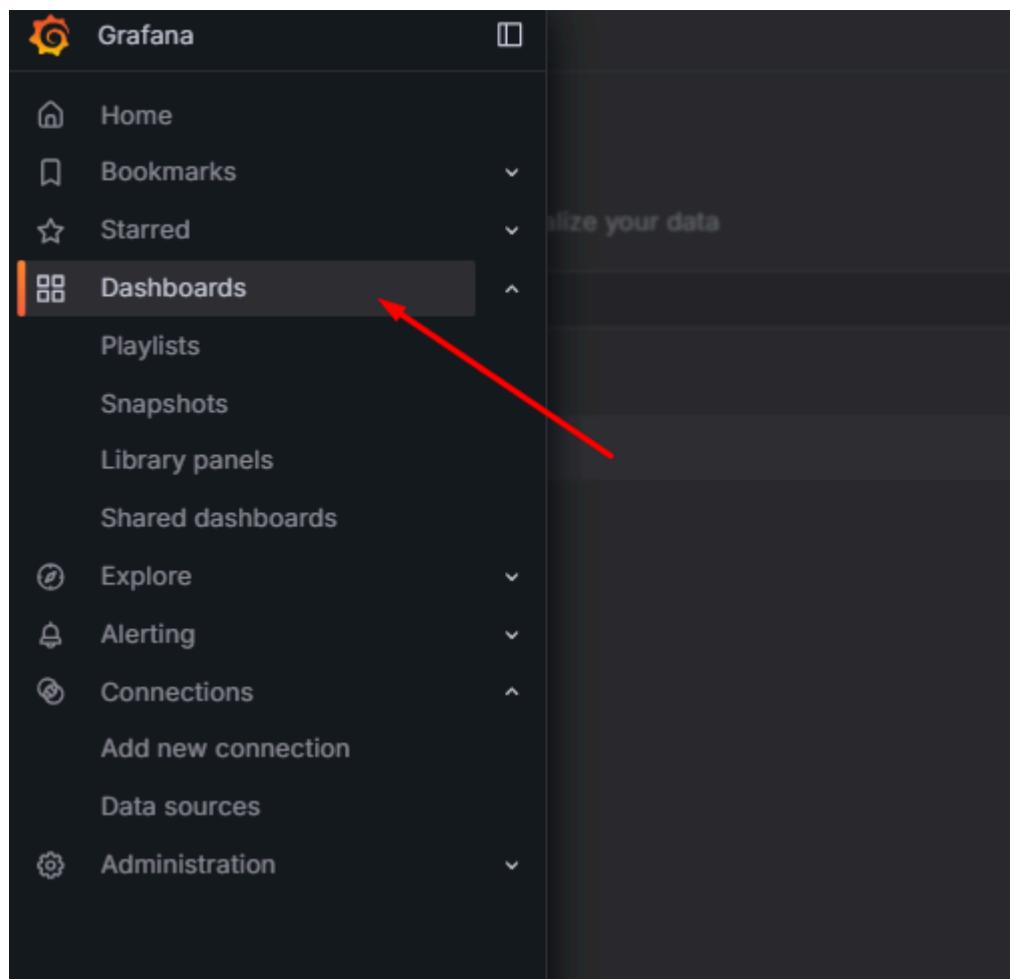
**Authentication**: Username is 'airbit', Password is 'configured' (marked as configured), and TLS/SSL Mode is 'disable'.

**Additional settings**: PostgreSQL Options include Version (15), Min time interval (1m), TimescaleDB (disabled), and Connection limits (Max open: 100, Auto max idle: checked, Max idle: 100, Max lifetime: 14400).

At the bottom are 'Delete' and 'Save & test' buttons.

## 5. Пример настройки виджета

### ▼ Шаг 1



▼ Шар 2

A screenshot of the Grafana Dashboards page. A red arrow points to the 'New dashboard' button in the top right corner of the header. The page shows a search bar, filter buttons, and a central area with a search icon and a message: 'No results found for your query'. There are also 'Clear search and filters' and 'Import' buttons.

▼ Шар 3

A screenshot of the 'Start your new dashboard by adding a visualization' page. A red arrow points to the '+ Add visualization' button. Below it, there are two sections: 'Import panel' with a '+ Add library panel' button, and 'Import a dashboard' with an 'Import dashboard' button.

▼ Шар 4

Выбираем созданный ранее источник данных, выбираем необходимую таблицу, задаем фильтрацию/группировку по необходимым параметрам

The screenshot shows the Grafana Query editor interface. At the top, it displays 'Queries 1', 'Transformations 2', and 'Alert 0'. Below this, the 'Data source' dropdown is set to 'grafana-postgresql-datasource'. A red arrow points to this dropdown. The 'Query options' section shows 'MD = auto = 1609' and 'Interval = 1m'. On the right, there's a 'Query inspector' button. The main area is titled 'A (grafana-postgresql-datasource)'. It includes sections for 'Format: Time series', 'Filter' (with a checked toggle), 'Group' (with a checked toggle), 'Order' (with a checked toggle), and 'Preview' (with a checked toggle). Red arrows point to the 'Filter', 'Group', 'Order', and 'Preview' toggles. Below these are 'Table' and 'Column' selection fields. The 'Table' field has 'parametrs' selected. A red arrow points to this selection. The 'Column' section has 'Choose' dropdowns for 'Column', 'Aggregation - optional', and 'Alias - optional'. There's also a 'Filter by column value - optional' section with a '+' button. The 'Order by' section allows selecting a column and setting ascending/descending order, with a limit of 50. A red arrow points to the 'Order by' section.

Например, выводим все параметры устройства с определенным DevEUI и упорядочиваем их по времени

This screenshot shows a detailed query configuration. It includes two columns: 'data' with aggregation 'Choose' and alias '"value"', and 'times' with aggregation 'Choose' and alias '"time"'. A filter for 'dev\_eui' is set to '3630313073386E19'. The 'Order by' section is set to 'times' with ascending order and a limit of 50. Red arrows point to the 'data' and 'times' sections.

Если необходимо вытащить конкретный параметр из JSON, выполняем шаг 5

Если grafana для столбца "times" не может назначить alias:"time", то добавляем этот столбец sq| запросом (Для этого необходимо переключить отображение нажатием из режима "Builder" в режим "Code"): SELECT data, times as time FROM sensor\_data LIMIT 50

## ▼ Шаг 5

Настраиваем согласно скриншоту, выбираем необходимые параметры устройства, например время и температуру

This screenshot shows the Grafana Transformations editor. It starts with a '1 - Extract fields' step where 'Source' is 'A value' and 'Format' is 'JSON'. A red arrow points to the 'Source' field. Under '1. Field', there's a placeholder 'A valid json path, e.g. "object.value1" or "object.value2[0]"' and a 'Add path' button. There are also 'Replace all fields' and 'Keep time' checkboxes. The second step, '2 - Filter fields by name', shows a 'From variable' dropdown with a radio button, an 'Identifier' dropdown with 'Regular expression pattern', and a 'Time' checkbox. Below these are buttons for 'Add another transformation' and 'Delete all transformations'. Red arrows point to the 'Time' checkbox and the 'Identifier' dropdown.

## 2) Установка и настройка HomeAssistant

1. Установить HomeAssistant командой:

```
docker run -d \
--name homeassistant \
--privileged \
--restart=unless-stopped \
-e TZ=MY_TIME_ZONE \
-v /homeassistant:/config \
-v /run/dbus:/run/dbus:ro \
--network=host \
ghcr.io/home-assistant/home-assistant:stable
```

2. Зайти и авторизоваться в HomeAssistant: <ip адрес локальной установки>:8123

Логин и пароль задаются при первом входе

3. Добавление поддержки MQTT

▼ Шаг 1

The screenshot shows the Home Assistant configuration interface. On the left, there's a sidebar with links: Обзор (Overview), Энергия (Energy), Журнал событий (Event Log), История (History), and Мультимедиа (Media). Below this is a 'Панель разработчика' (Developer Panel) with tabs: Настройки (Settings) (which is selected and highlighted in blue), Уведомления (Notifications), and airpi. The main area is titled 'Настройки' (Settings) and contains a list of configuration options:

- Home Assistant Cloud (Удалённый доступ к серверу, интеграция с Alexa и Google Assistant)
- Устройства и службы (Интеграции, устройства и объекты)
- Автоматизация и сцены (Сцены, автоматизация, скрипты и их проекты)
- Пространства, зоны и ярлыки (Места внутри и снаружи вашего дома)
- Панели (Визуальное взаимодействие с вашим домом)
- Голосовые ассистенты (Управление виртуальными голосовыми ассистентами)
- Метки (Настройка меток NFC и QR-кодов)
- Люди (Люди, которые имеют доступ к вашему дому)
- Система (Резервные копии, журналы, перезагрузка системы)
- О программе (Информация об используемой версии)

A red arrow points from the text 'Добавление поддержки MQTT' to the 'Интеграции' (Integrations) link at the top of the main settings list.

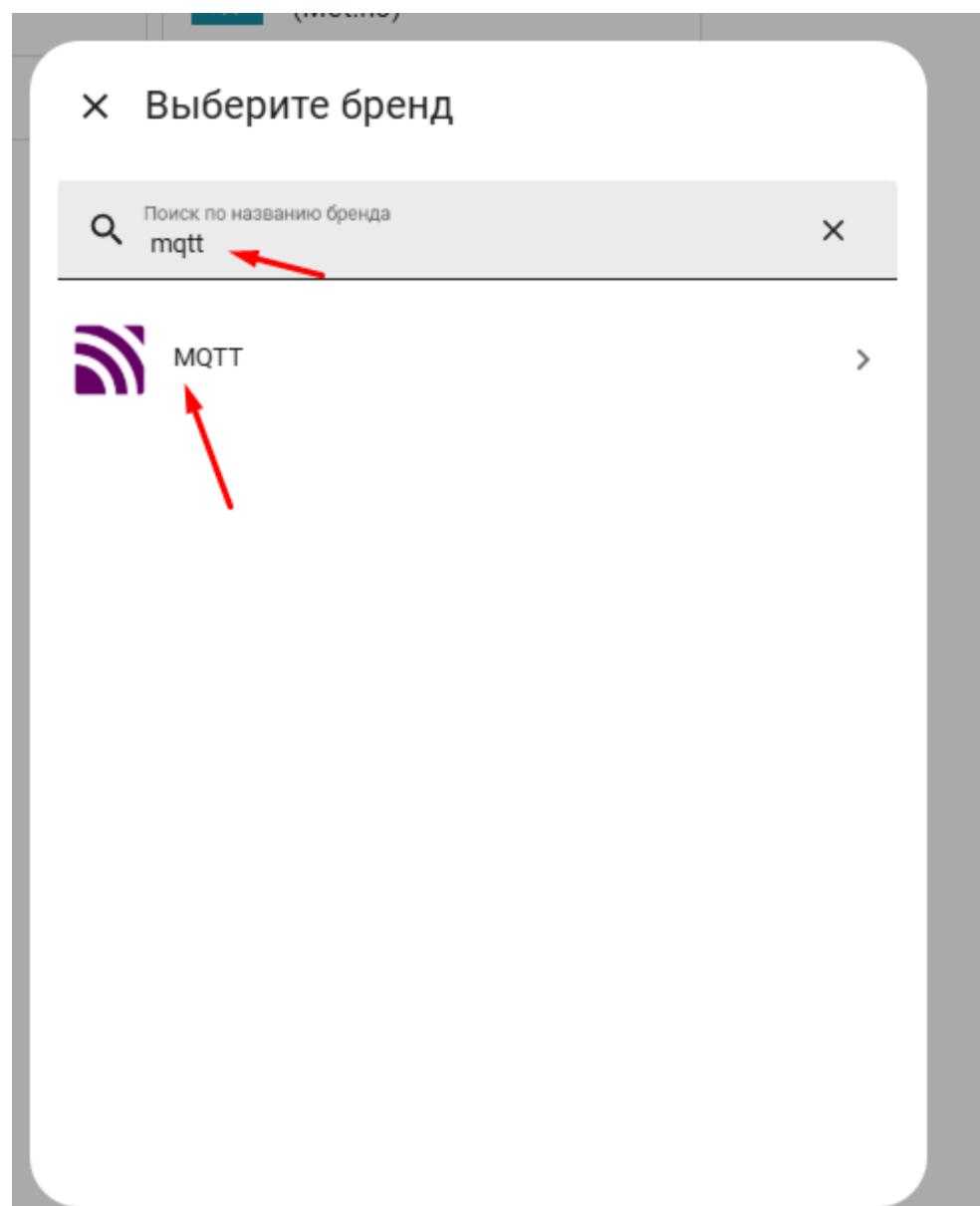
▼ Шаг 2

The screenshot shows the 'Интеграции' (Integrations) page. The sidebar and developer panel are identical to the previous screenshot. The main area displays three integration cards:

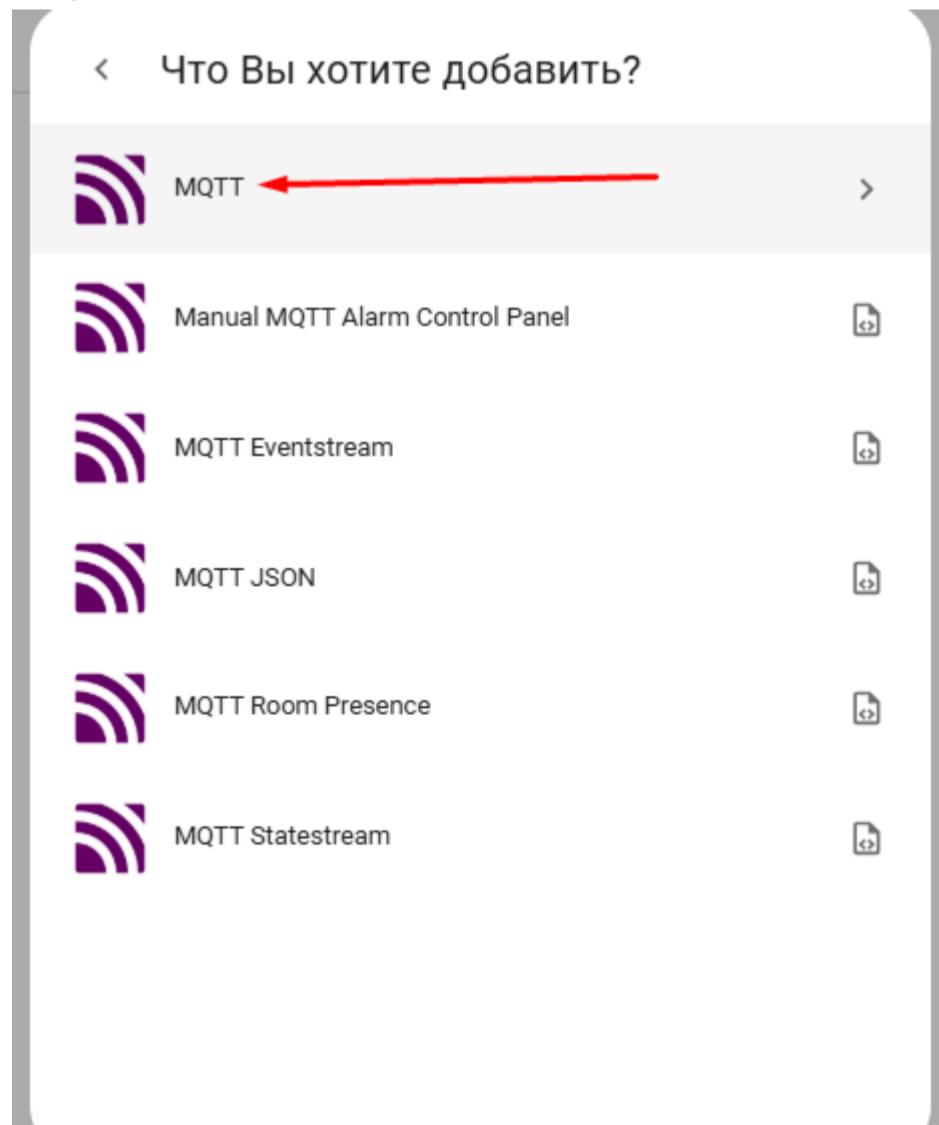
- Bluetooth (1 УСТРОЙСТВО)
- Google Translate text-to-speech (1 ОБЪЕКТ)
- Meteorologisk institutt (Met.no) (1 СЛУЖБА)

A red arrow points from the text 'Добавление поддержки MQTT' to the 'Добавить интеграцию' (Add Integration) button at the bottom right of the screen.

▼ Шаг 3

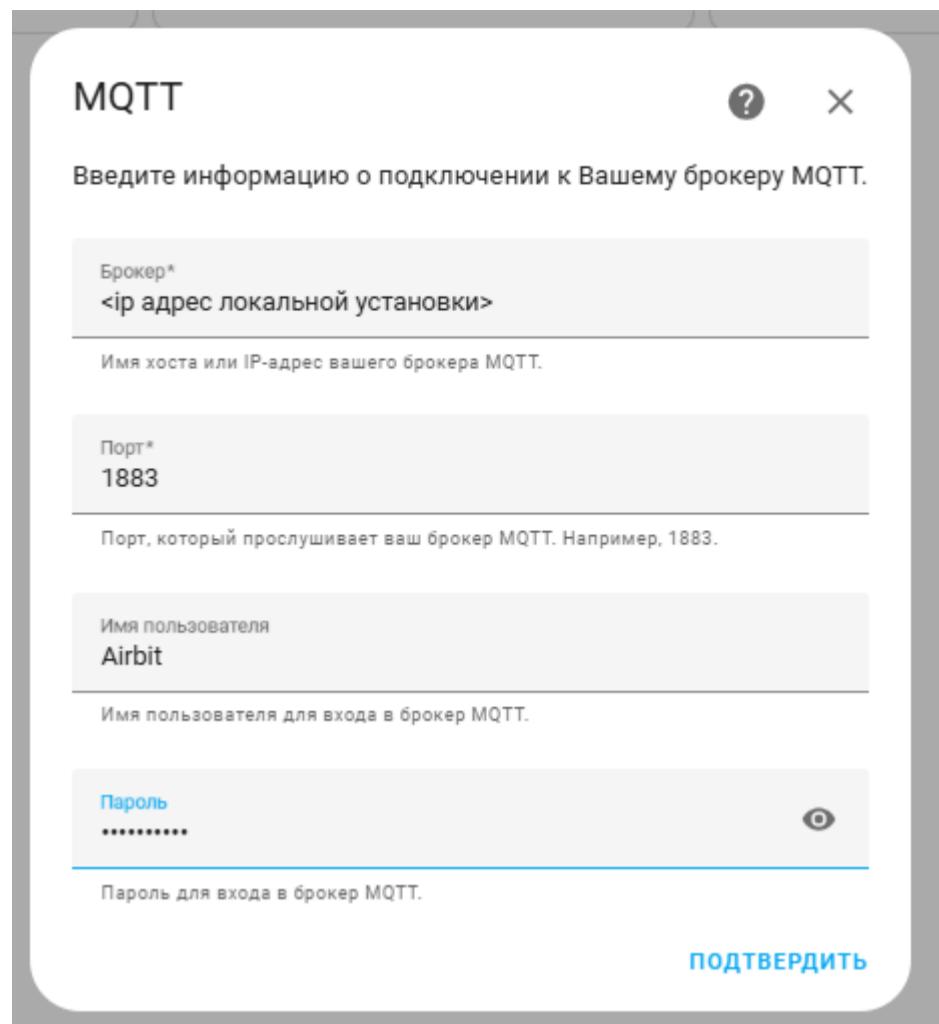


▼ Шаг 4



▼ Шаг 5

Логин и пароль аналогичные тем, которые задавали для MQTT брокера



#### ▼ Шаг 6

Добавляем в файл конфига MQTT, по умолчанию конфиг находится в директории указанной в пункте 1

```
# Loads default set of integrations. Do not remove.
default_config:

# Load frontend themes from the themes folder
frontend:
  themes: !include_dir_merge_named themes

# Text to speech
tts:
  - platform: google_translate

automation: !include automations.yaml
script: !include scripts.yaml
scene: !include scenes.yaml
modbus: !include modbus.yaml
sensor: !include sensor.yaml
mqtt: !include mqtt.yaml
media_player: !include media_player.yaml
telegraf_bots:
```

#### 4. Добавление устройства

##### ▼ Пример конфигурации yaml файла

В той же директории, где находится общий конфиг создаем файл "mqtt.yaml", настраиваем получение параметров из топиков

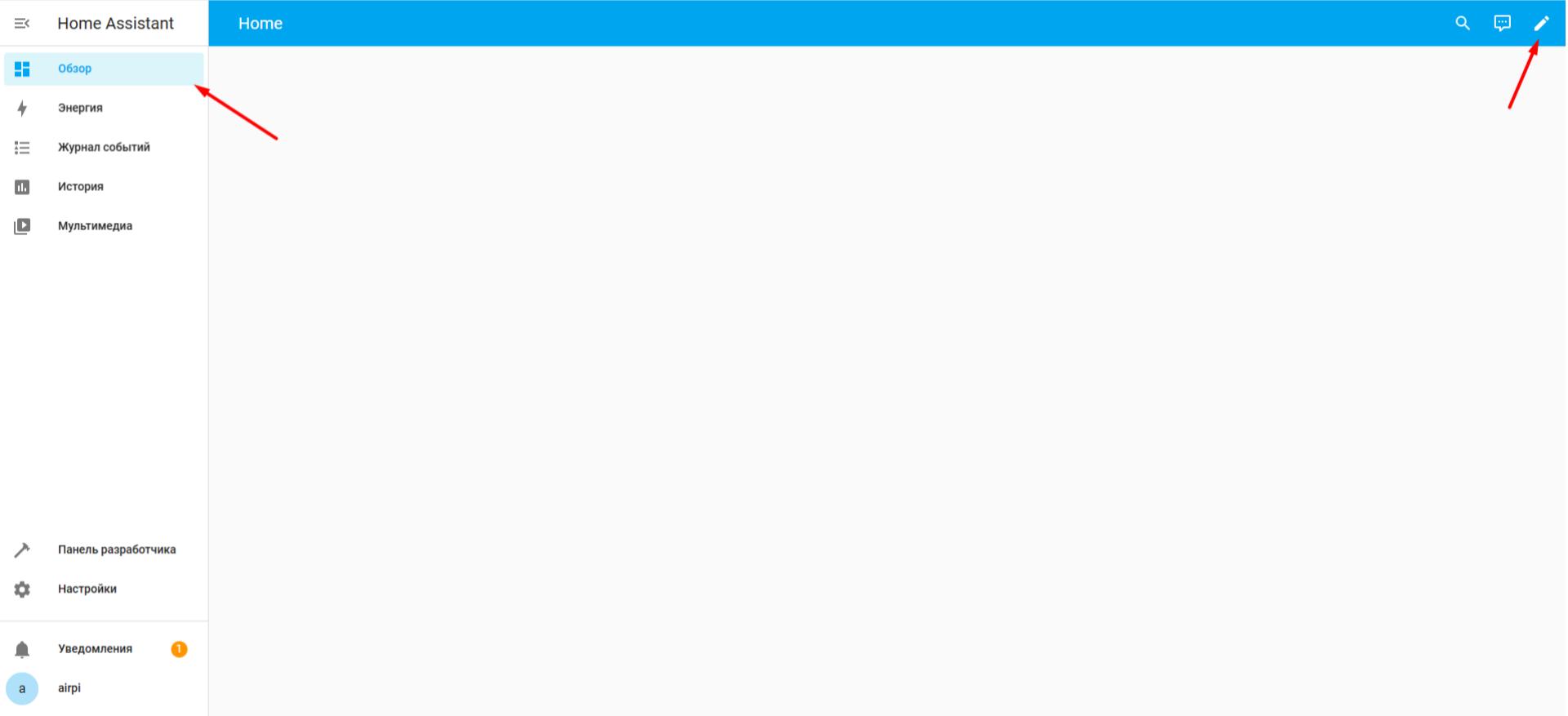
```
sensor:
  - name: "Um_temp"
    unique_id: um_temp_id
    state_topic: "ha/3630313073386E19"
    suggested_display_precision: 1
    unit_of_measurement: "°C"
    value_template: "{{value_json.data[0].temp}}".
  - name: "Um_lux"
    unique_id: um_lux_id
    state_topic: "ha/3630313073386E19"
    suggested_display_precision: 1
    unit_of_measurement: "lux"
    value_template: "{{value_json.data[0].lux}}"
  - name: "Um_co2"
    unique_id: um_co_id
    state_topic: "ha/3630313073386E19"
    suggested_display_precision: 1
    unit_of_measurement: "ppm"
    value_template: "{{value_json.data[0].co2}}"
  - name: "Um_hum"
    unique_id: um_hum_id
    state_topic: "ha/3630313073386E19"
    suggested_display_precision: 1
    unit_of_measurement: "%"
    value_template: "{{value_json.data[0].hum}}"
```

Если устройства на сетевом сервере добавлены в группу "Grafana", то данные будут поступать в топики: grafana/{devEUI\_устройства}

## 5. Пример добавления дашборда

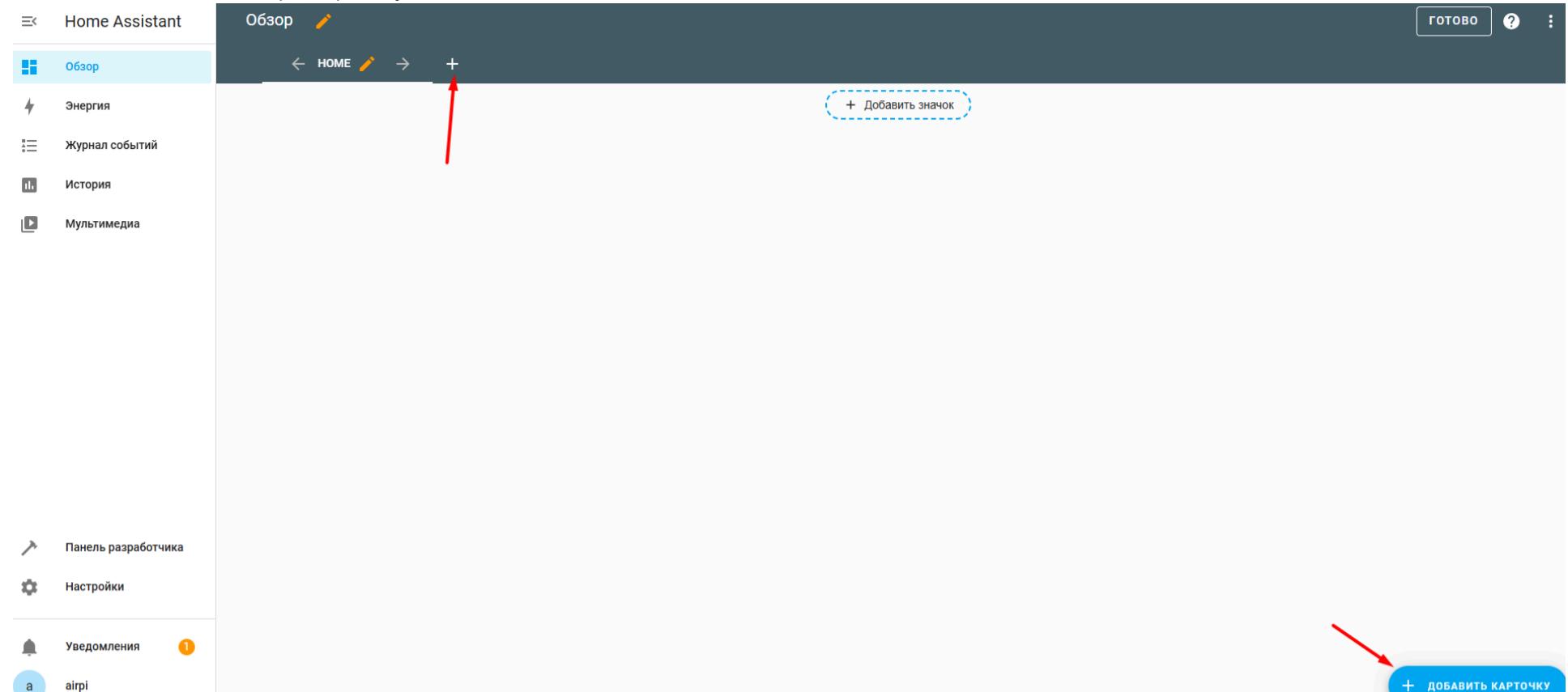
### ▼ Шаг 1

Переходим в режим редактирования



### ▼ Шаг 2

Добавляем новый дашборд/карточку



▼ Шаг 3

Выбираем тип карточки, например: "объект"

The dialog title is 'Какую карточку Вы хотели бы добавить в раздел "Home"?' (Which card do you want to add to the "Home" section?). It has two tabs: 'КАРТОЧКИ' (Cards) and 'ОБЪЕКТЫ' (Objects). The 'КАРТОЧКИ' tab is active. A search bar says 'Поиск карточек'. Below are several card types:

- Карта**: Позволяет отображать объекты на карте.
- Кнопка**: Используется для отображения в интерфейсе медиаплеера с простыми в использовании элементами управления.
- Мультимедиа**: Используется для отображения в интерфейсе медиаплеера с простыми в использовании элементами управления.
- Объект**: Sun Следующий рассвет 28 марта 2025 г...
- Объекты**: Sun Следующий рассвет Через 17 часов, Sun Следующий рассвет Через 7 часов, Sun Следующий рассвет Через 12 часов.
- Освещение**: Позволяет контролировать источник света, изменять яркость.
- Панель сигнализации**: Позволяет управлять панелью сигнализации, открыть объекты и т.д.
- Плитка**: Sun Следующий рассвет
- Прогноз погоды**: Облачно

A red arrow points from the text 'Выбираем тип карточки' to the 'Объект' card. Another red arrow points from the text 'Например: "объект"' to the 'Объекты' card. A blue 'ОТМЕНИТЬ' (Cancel) button is at the bottom right.

▼ Шаг 4

Выбираем вычитываемый из топиков параметр, настраиваем визуализацию и сохраняем

Настройка карточки "Объект"

НАСТРОЙКИ ВИДИМОСТЬ

Объект\* Um\_lux

Название Освещенность

Атрибут Unit of measurement

Единица измерения

Тема (необязательно)

Показывать цвет состояния

ОТМЕНИТЬ СОХРАНИТЬ

## Создание образа настроенной системы

Создание образа:

1. Создать образ системы:

```
sudo dd if=<путь до карты памяти> of=<название файла>.img bs=1M status=progress
```

При необходимости уменьшить параметр bs и/или сжать готовый образ согласно пункту 2

2. Установить pishrink командой:

```
sudo apt install pishrink
```

или

```
wget https://raw.githubusercontent.com/Drewsif/PiShrink/master/pishrink.sh  
chmod +x pishrink.sh  
sudo mv pishrink.sh /usr/local/bin/pishrink
```

Сжать образ командой:

```
sudo pishrink <путь до файла, его название>.img
```

3. Взять карту памяти, сбросить на неё файловую систему. Например при помощи "Управление дисками" в Windows
4. Скопировать образ:

```
sudo dd if=<путь до файла, его название>.img of=<путь до карты памяти> bs=1M status=progress
```

При необходимости уменьшить параметр bs

## Первый запуск

1. Стандартное логин: [airpi@air-bit.ru](mailto:airpi@air-bit.ru) пароль: airpi62
2. Если у Вас нет лицензии, то в открывшемся окне необходимо нажать на кнопку скачивания файла лицензии и прислать её на почту [support@air-bit.ru](mailto:support@air-bit.ru) (Также приложите карточку компании)

После получения ответного файла лицензии, необходимо его загрузить и активировать на той же странице

3. Создать хотя бы одну сеть (Раздел "Настройки" -> "Сети")
4. Создать хотя бы один частотный план (Раздел "Настройки" -> "Частотный план")