

Лабораторная работа 2. Практика по Docker

Выполнила студент гр. 5142704/30801 Гайна А. А.

Шаг 1: Разработка Simulator для генерации данных

В файле *sensor.py* создаем четвертый тип датчиков CO, прназначение которого измерять концентарцию угарного газа.

```
class CO(Sensor):
    step = 0

    def __init__(self,name):
        super().__init__(name)
        self.type = "carbon oxid"

    def generate_new_value(self):
        self.value = self.step * 1e6
        self.step = self.step + 0.001
```

Далее в файле *main.py* реализуем клиента, который подключается к mqtt брокеру и публикует сообщения.

```
import paho.mqtt.client as paho
from os import environ
import time
```

```
from entity.sensor import *
```

```
broker = "localhost" if "SIM_HOST" not in environ.keys() else environ["SIM_HOST"]
port = 1883 if "SIM_PORT" not in environ.keys() else environ["SIM_PORT"]
name = "sensor" if "SIM_NAME" not in environ.keys() else environ["SIM_NAME"]
period = 1 if "SIM_PERIOD" not in environ.keys() else int(environ["SIM_PERIOD"])
type_sim = "temperature" if "SIM_TYPE" not in environ.keys() else environ["SIM_TYPE"]

sensors = {"temperature": Temperature, "pressure": Pressure, "current": Current, "carbon_oxid":
CO}
```

```

def on_publish(client, userdata, result): # create function for callback
    print(f'data published {userdata}')
    pass

sensor = sensors[type_sim](name=name)
client1 = paho.Client(sensor.name) # create client object
client1.on_publish = on_publish # assign function to callback
client1.connect(broker, port) # establish connection
while True:
    sensor.generate_new_value()
    ret = client1.publish("sensors/" + sensor.type + "/" + sensor.name, sensor.get_data()) # publish
    time.sleep(period)

```

В *Dockerfile*, необходимом для того, что создать образ, указываем следующие инструкции:

```

FROM python:alpine3.19
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["python", "main.py"]

```

Инструкция FROM инициализирует новый этап сборки и устанавливает базовый образ для последующих инструкций. WORKDIR создает рабочий каталог для последующих инструкций Dockerfile. Инструкция COPY копирует файл requirements.txt из источника в указанное место внутри образа.

Инструкция RUN задает команды, которые следует выполнить и поместить в новый образ контейнера. RUN описывает команду с аргументами, которую нужно выполнить когда контейнер будет запущен.

Содержимое файла *requirements.txt* приведено ниже:

```
paho_mqtt==1.6.1
```

paho_mqtt предоставляет клиентский класс, который позволяет приложениям подключаться к MQTT-брокеру для публикации сообщений.

После сделанных выше операций можно создать образ командой:

```
docker build -t annagajna/data-simulator .
```

Таким образом создан образ, из которого можно развернуть контейнер.

View a summary of image vulnerabilities and recommendations → docker scout quickview > docker images				
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
annagajna/data-simulator	latest	9d115dcf5643	About a minute ago	57.5MB

Шаг 2: Запуск Mosquitto брокера

Для настройки протокола MQTT необходимо создать конфигурационный файл *mosquitto.conf* со следующим содержанием:

```
listener 1883
```

```
allow_anonymous true
```

Для более удобного запуска брокера создадим файл *docker-compose.yml* со следующим содержанием:

```
version: "3"
```

```
services:
```

```
  broker:
```

```
    image: eclipse-mosquitto
```

```
    container_name: broker
```

```
    volumes:
```

```
      - ./mosquitto/mosquitto.conf:/mosquitto/config/mosquitto.conf
```

```
    ports:
```

```
      - "1883:1883"
```

Теперь создав контейнер (из Шага 1), получим следующее:

```
C:\Документы\магистратура\перевод\облачные сервисы\dockerpractice\client\simulator>docker run -e SIM_HOST=192.168.0.101 -e SIM_TEMP=temperature --name test -d annagajna/data-simulator
5b996c54e17cd48b5ccc37f03107da1816c256d4fe05a00af5bae1c58ce5f6df
```

Брокер отображает присоединившегося клиента.

```
PS C:\Документы\магистратура\перевод\облачные сервисы\dockerpractice\gateway> docker compose up
time="2024-09-20T16:23:16+03:00" level=warning msg="C:\\Документы\\магистратура\\перевод\\облачные сервисы\\dockerpractice\\gateway\\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 1/0
✔ Container broker Created 0.0s
Attaching to broker
broker | 1726838597: mosquitto version 2.0.18 starting
broker | 1726838597: Config loaded from /mosquitto/config/mosquitto.conf.
broker | 1726838597: Opening ipv4 listen socket on port 1883.
broker | 1726838597: Opening ipv6 listen socket on port 1883.
broker | 1726838597: mosquitto version 2.0.18 running
```

Теперь можно запустить несколько датчиков. Но перед этим необходимо прописать *docker-compose.yml*:

```
version: "3"
```

```
services:
```

```
  temp_sensor:
```

```
    image: annagajna/data-simulator
```

```
    environment:
```

- SIM_HOST=192.168.0.101
- SIM_NAME=TEMP1
- SIM_PERIOD=2
- SIM_TYPE=temperature

```
  pressure_sensor:
```

```
    image: annagajna/data-simulator
```

```
    environment:
```

- SIM_HOST=192.168.0.101
- SIM_NAME=PRESS1
- SIM_PERIOD=2
- SIM_TYPE=pressure

```
  current_sensor:
```

```
    image: annagajna/data-simulator
```

```
    environment:
```

- SIM_HOST=192.168.0.101
- SIM_NAME=CURRENT1
- SIM_PERIOD=2
- SIM_TYPE=current

```
  co_sensor:
```

```
    image: annagajna/data-simulator
```

```
    environment:
```

- SIM_HOST=192.168.0.101
- SIM_NAME=CO1
- SIM_PERIOD=2
- SIM_TYPE=carbon_oxid

```
  temp_sensor_2:
```

```
    image: annagajna/data-simulator
```

environment:

- SIM_HOST=192.168.0.101
- SIM_NAME=TEMP2
- SIM_PERIOD=4
- SIM_TYPE=temperature

co_sensor_2:

image: annagajna/data-simulator

environment:

- SIM_HOST=192.168.0.101
- SIM_NAME=CO2
- SIM_PERIOD=6
- SIM_TYPE=carbon_oxid

С помощью команды 'docker compose up' запустим одновременно 6 контейнеров:

```
PS C:\Документы\магистратура\перевод\облачные сервисы\dockerPractice\vm\client\simulator> docker compose up
time="2024-09-24T18:04:35+03:00" level=warning msg="C:\\Документы\\магистратура\\перевод\\облачные сервисы\\dockerPractice\\vm\\client\\simulator\\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 7/7
  ✓ Network simulator_default          Created                                0.0s
  ✓ Container simulator-pressure_sensor-1 Created                                0.1s
  ✓ Container simulator-current_sensor-1 Created                                0.1s
  ✓ Container simulator-co_sensor_2-1  Created                                0.1s
  ✓ Container simulator-temp_sensor_2-1 Created                                0.1s
  ✓ Container simulator-temp_sensor-1  Created                                0.1s
  ✓ Container simulator-co_sensor-1    Created                                0.1s
Attaching to co_sensor-1, co_sensor_2-1, current_sensor-1, pressure_sensor-1, temp_sensor-1, temp_sensor_2-1
```

Шаг 3: Получение данных от симулятора

Первоначально необходимо настроить Telegraf, который подписывается на MQTT, где датчики публикуют данные. Данные будут сохраняться в InfluxDB. Отображение информации с датчиков будет происходить при помощи Grafana.

Telegraf

Перед использованием Telegram необходимо настроить его.

Для этого откроем конфигурационный файл *telegraf.conf* и в разделе `[[inputs.mqtt_consumer]]` пропишем следующее:

```
servers = ["tcp://192.168.0.101:1883"] # адрес vm с mqtt-брокером
topics = [
    "sensors/#"
]
data_format = "value"
data_type = "float"
```

Тем самым мы настраиваем Telegraf на чтение данных с машины IP адрес которой 192.168.0.101 через порт 1883.

Также настроим раздел вывода `[[outputs.influxdb]]`:

```
urls = ["http://influxdb:8086"]
```

```
database = "sensors"
```

```
skip_database_creation = true
```

```
username = "telegraf"
```

```
password = "telegraf"
```

Настройка Telegraf на этом завершена.

InfluxDB

Для создания базы данных необходимо в конфигурационном файле *influxdb-init.iql* прописать следующее:

```
CREATE database sensors
```

```
CREATE USER telegraf WITH PASSWORD 'telegraf' WITH ALL PRIVILEGES
```

Grafana

Данные для отображения датчиков берутся из InfluxDB.

Для настройки в конфигурационном файле необходимо прописать следующее:

```
apiVersion: 1
```

```
datasources:
```

```
- name: InfluxDB_v1
```

```
  type: influxdb
```

```
  access: proxy
```

```
  database: sensors
```

```
  user: telegraf
```

```
  url: http://influxdb:8086
```

```
  jsonData:
```

```
    httpMode: GET
```

```
  secureJsonData:
```

```
    password: telegraf
```

Для запуска всех трех контейнеров воспользуемся `docker-compose`.

В *docker-compose.yml* пропишем следующее:

version: "3"

services:

influxdb:

image: influxdb:1.8

container_name: influxdb

volumes:

- ./influxdb/scripts:/docker-entrypoint-initdb.d
- influx_data:/var/lib/influxdb

networks:

- server-net

telegraf:

image: telegraf

container_name: telegraf

volumes:

- ./telegraf:/etc/telegraf:ro

restart: unless-stopped

networks:

- server-net

grafana:

image: grafana/grafana

container_name: grafana

volumes:

- grafana_data:/var/lib/grafana
- ./grafana:/etc/grafana/

environment:

- GF_SECURITY_ADMIN_USER=admin
- GF_SECURITY_ADMIN_PASSWORD=admin
- GF_USERS_ALLOW_SIGN_UP=false

restart: unless-stopped

ports:

- 3000:3000

networks:

- server-net

volumes:

influx_data: {}

grafana_data: {}

networks:

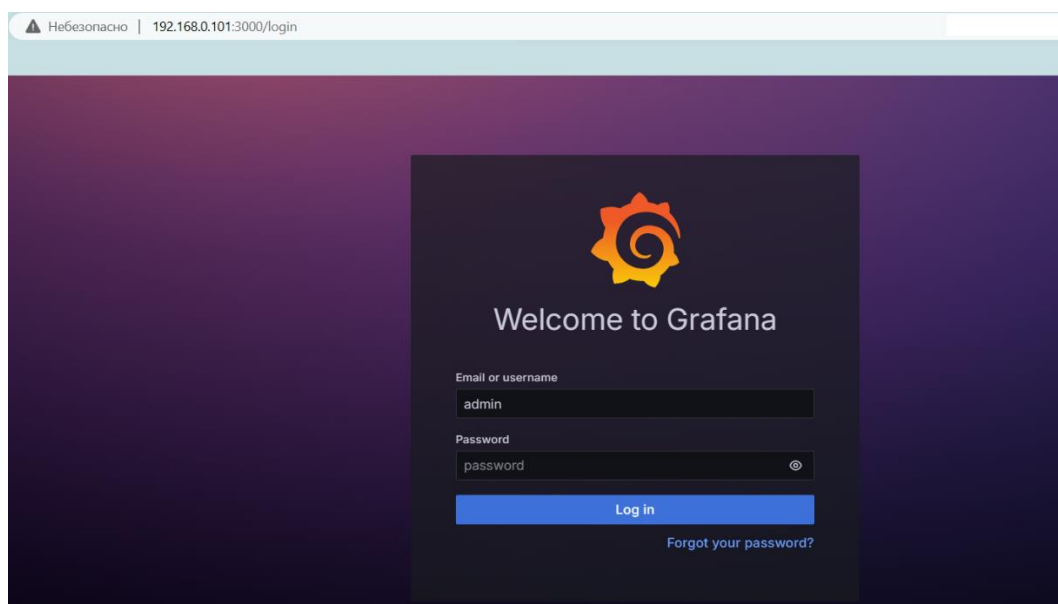
server-net: {}

После чего можно выполнить команду 'docker compose up'

```
Windows PowerShell
PS C:\Документы\магистратура\перевод\облачные сервисы\dockerPractice\vm\server\infra\telegraf> docker compose up
time="2024-09-21T18:20:38+03:00" level=warning msg="C:\\Документы\\магистратура\\перевод\\облачные сервисы\\dockerPractice\\vm\\server\\infra\\telegraf\\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 19/19
  ✓ influxdb Pulled
  ✓ ba83bbfca944 Pull complete
  ✓ 48e77900ed2 Pull complete
  ✓ d32c4e3b42ad Pull complete
  ✓ f51177c502b7 Pull complete
  ✓ 2e0cc17e692b Pull complete
  ✓ celac99f35a4 Pull complete
  ✓ 048078e3cfbf Pull complete
  ✓ grafana Pulled
  ✓ 4abcf2066143 Pull complete
  ✓ 39aee5fd3406 Pull complete
  ✓ 592f1e71407c Pull complete
  ✓ 66aec874ce0c Pull complete
  ✓ bde37282dfba Pull complete
  ✓ b6982d0733af Pull complete
  ✓ ab3c28da242b Pull complete
  ✓ e4892977d944 Pull complete
  ✓ ef2b3f3f597e Pull complete
  ✓ 27a3c8ebdfbf Pull complete
[+] Running 6/6
  ✓ Network telegraf_server-net Created
  ✓ Volume "telegraf_grafana_data" Created
  ✓ Volume "telegraf_influx_data" Created
  ✓ Container grafana Created
  ✓ Container influxdb Created
  ✓ Container telegraf Created
Attaching to grafana, influxdb, telegraf
grafana | GF_PATHS_CONFIG='/etc/grafana/grafana.ini' is not readable.
```

Настройка дашборда

После запуска всех необходимых контейнеров, в браузер переходим по '192.168.0.101:3000'



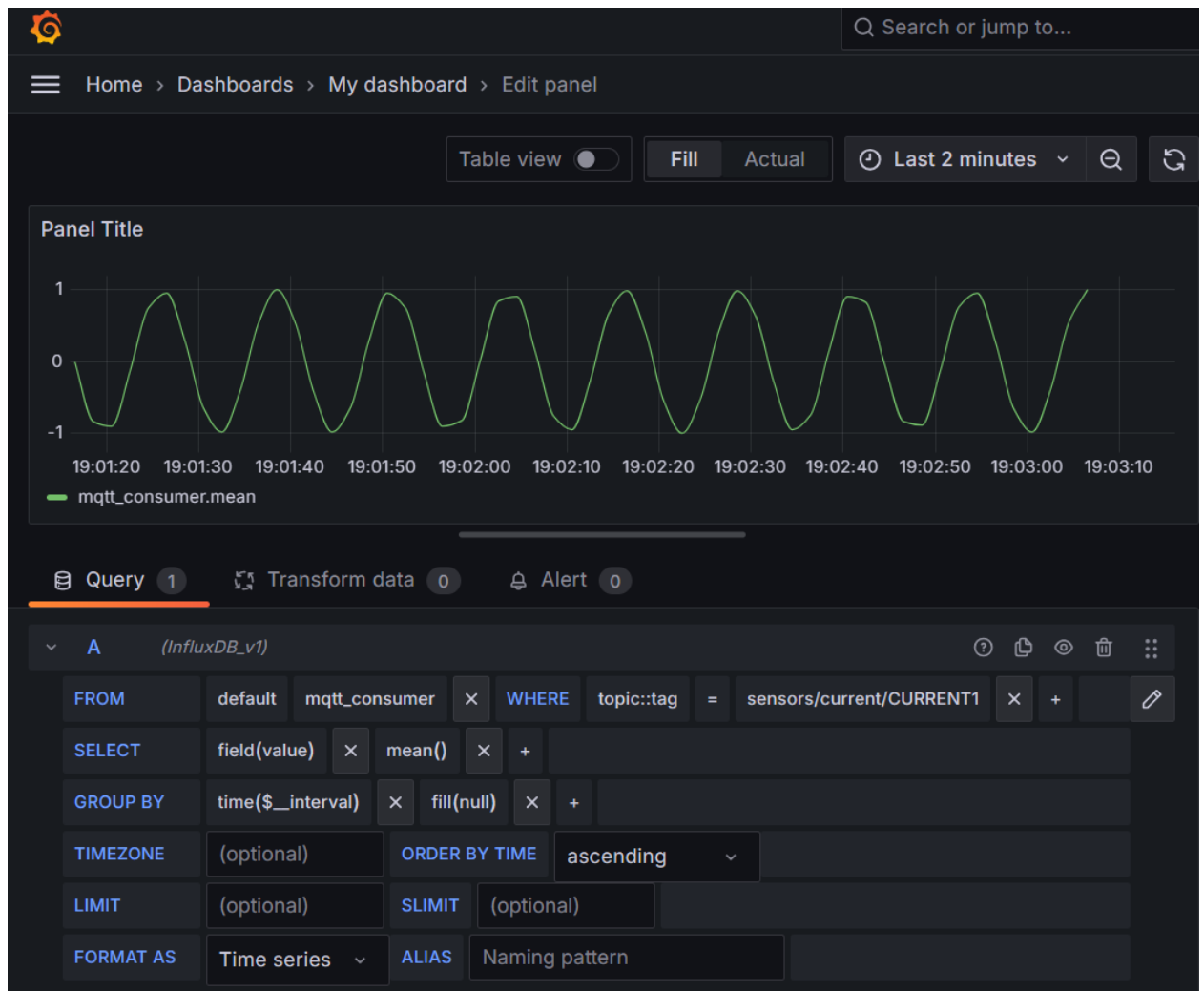
Для проверки соединения перейдем в Menu -> Connections -> Data sources. Находим в источниках InfluxDB и проверяем подключение:

To support data isolation and security, make sure appropriate permissions are configured in InfluxDB.

Database	sensors	
User	telegraf	
Password	configured	Reset
HTTP Method	GET	
Min time interval	10s	
Max series	1000	

Testing... this could take up to a couple of minutes

После переходим через Меню в раздел Dashboards, где создаем собственный дашбоард. Для отображения информации с датчиков необходимо создать запрос (query).



После создания необходимо количества графиков, отображающих информацию с Simulator, экспортируем дашборд как JSON-файл.

Для этого находим функцию Share -> Export -> Save to file. Сохраненный файл помещаем в папку vms\server\infra\grafana\provisioning\dashboards\mqtt.json

Пример выполненной работы

