Persiapan

Instalasi software

1. Install miniconda
2. Install go
3. Install Mosquitto
4. Install InfluxDB
5. Install telegraf
6. Install Grafana

Membuat Environment Python 3.10

conda create -n py310-iot python=3.10

Setelah itu, aktifkan dengan:

conda activate py310-iot

Instalasi Paket-paket Python

conda install numpy pandas paho-mqtt

Memasukkan data dari sensor ke MQTT Broker / Server

Ringkasan:

Membuat software pada peranti IoT untuk mem-*publish* data ke MQTT Broker. Software tersebut mempunyai fungsionalitas yang sama dengan **mosquitto_pub**. Data ini nantinya akan diambil oleh **telegraf** dan dimasukkan ke **InfluxDB**

1. Hidupkan MQTT Broker (Mosquitto)

```
$ mosquitto
1663337304: mosquitto version 2.0.15 starting
1663337304: Using default config.
1663337304: Starting in local only mode. Connections will only be possible from
clients running on this machine.
1663337304: Create a configuration file which defines a listener to allow
remote access.
1663337304: For more details see
https://mosquitto.org/documentation/authentication-methods/
1663337304: Opening ipv4 listen socket on port 1883.
1663337304: Opening ipv6 listen socket on port 1883.
```

```
1663337304: mosquitto version 2.0.15 running
```

Untuk memeriksa bahwa broker sudah berjalan, coba mekanisme pub/sub berikut ini (lakukan pada 2 terminal/shell/window powershell):

Subscribe

```
$ mosquitto_sub -h localhost -t 'sensors/temperature' -v
```

Publish

```
$ mosquitto_pub -h localhost -t 'sensors/temperature' -m '22'
```

Setelah mosquitto_pub di atas dikerjakan, maka pada terminal/shell/window untuk Subscribe akan muncul berikut ini:

```
$ mosquitto_sub -h localhost -t 'sensors/temperature' -v
sensors/temperature 22
```

Menggunakan Python

Jalankan program untuk mem*publish* data ke MQTT Broker.

```
$ python simple-sensor.py -b localhost -v -i 5
starting
Publishing on  sensors/sensor-3542
send control to  sensors/sensor-3542/control
Sensors States are  ['1', '0']
connecting to broker localhost:1883
Attempts  0
publish on sensors/sensor-3542  message   0
publish on sensors/sensor-3542  message   0
…
…
```

Pada window lain, buka untuk subscribe ke data yang dipublish setiap 5 detik tersebut:

```
$ mosquitto_sub -h localhost -t 'sensors/sensor-3542' -v
sensors/sensor-3542 0
sensors/sensor-3542 0
…
….
sensors/sensor-3542 1
sensors/sensor-3542 1
```

```
sensors/sensor-3542 1
…
…
```

Jika ingin mengatur sensor, bisa diberikan perintah menggunakan **mosquitto_pub** berikut:

```
mosquitto_pub -h localhost -t sensors/sensor-3542/control -m 1
```

Hasilnya adalah:

```
$ python simple-sensor.py -b localhost -v -i 5
starting
Publishing on  sensors/sensor-3542
send control to  sensors/sensor-3542/control
Sensors States are  ['1', '0']
connecting to broker localhost:1883
Attempts  0
publish on sensors/sensor-3542  message   0
publish on sensors/sensor-3542  message   0
…
…
control message  1
updating status 1
publish on sensors/sensor-3542  message   1
publish on sensors/sensor-3542  message   1
```

Untuk subscriber:

```
…
…
sensors/sensor-3542 1
sensors/sensor-3542 1
sensors/sensor-3542 1
```

Menggunakan Go

Build:

```
$ go build mqtt.go
$ ls -la
-rw-r--r-- 1 bpdp bpdp     211 Oct 28  2021 go.mod
-rw-r--r-- 1 bpdp bpdp     975 Oct 28  2021 go.sum
-rwxr-xr-x 1 bpdp bpdp 6591293 Sep 17 04:12 mqtt*
-rw-r--r-- 1 bpdp bpdp    1204 Oct 28  2021 mqtt.go
```

Jalankan *executable* yang dihasilkan:

```
./mqtt
2022-09-17 04:21:30.810043753 +0700 WIB m=+1.001038916
2022-09-17 04:21:31.810164331 +0700 WIB m=+2.001159494
2022-09-17 04:21:32.810150221 +0700 WIB m=+3.001145384
2022-09-17 04:21:33.810053415 +0700 WIB m=+4.001048584
2022-09-17 04:21:34.810099343 +0700 WIB m=+5.001094506
2022-09-17 04:21:35.810055682 +0700 WIB m=+6.001050844
2022-09-17 04:21:36.810251177 +0700 WIB m=+7.001246348
2022-09-17 04:21:37.810075898 +0700 WIB m=+8.001071062
2022-09-17 04:21:38.810062155 +0700 WIB m=+9.001057331
…
…
```

Catatan: jika di Windows, langsung panggil **mqtt**, tidak perlu **./mqtt**

Aktifkan subscriber:

```
$ mosquitto_sub -h localhost -t 'sensors' -v
sensors weather,location=gedung-mti temperature=61 1663363390331399120
sensors weather,location=gedung-mti temperature=63 1663363391331641292
sensors weather,location=gedung-mti temperature=66 1663363392330886701
sensors weather,location=gedung-mti temperature=91 1663363393331074462
sensors weather,location=gedung-mti temperature=47 1663363394331320825
sensors weather,location=gedung-mti temperature=38 1663363395331416032
sensors weather,location=gedung-mti temperature=79 1663363396331642754
sensors weather,location=gedung-mti temperature=22 1663363397331274914
…
…
```

Mengaktifkan InfluxDB

InfluxDB bisa diaktifkan dengan menjalankan daemon:

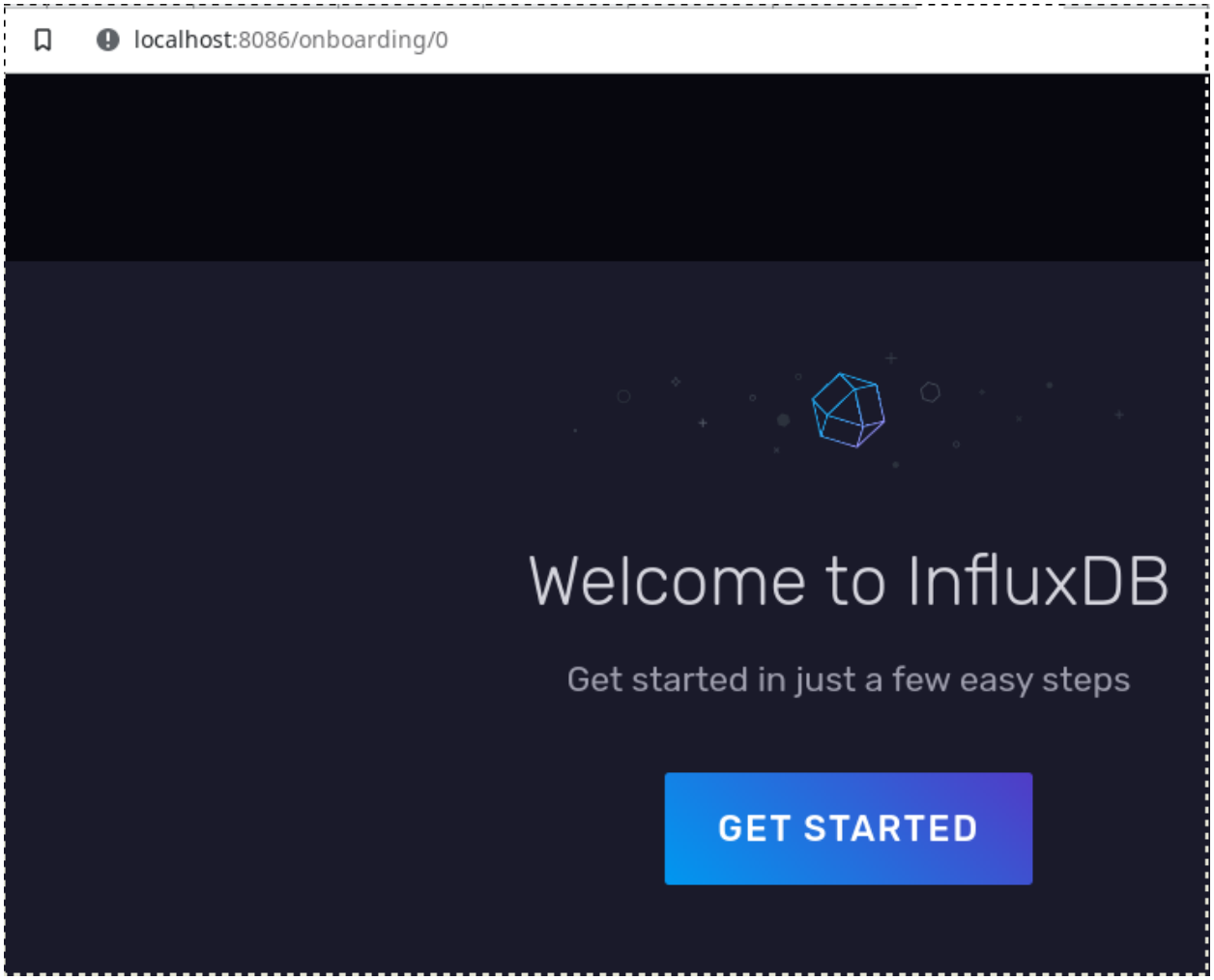
```
$ influxd
2022-09-16T16:14:26.046426Z    info  Welcome to InfluxDB      {"log_id":
"0cym7M~W000", "version": "v2.4.0", "commit": "de247bab08", "build_date":
"2022-08-18T19:41:15Z", "log_level": "info"}
2022-09-16T16:14:26.199649Z    info  Resources opened  {"log_id":
"0cym7M~W000", "service": "bolt", "path":
"/home/bpdp/.influxdbv2/influxd.bolt"}
2022-09-16T16:14:26.199969Z    info  Resources opened  {"log_id":
"0cym7M~W000", "service": "sqlite", "path":
"/home/bpdp/.influxdbv2/influxd.sqlite"}
```

```
2022-09-16T16:14:26.221904Z    info  Bringing up metadata migrations
{"log_id": "0cym7M~W000", "service": "KV migrations", "migration_count": 20}
2022-09-16T16:14:31.378781Z    info  Bringing up metadata migrations
{"log_id": "0cym7M~W000", "service": "SQL migrations", "migration_count": 7}
2022-09-16T16:14:32.366828Z    info  Using data dir    {"log_id":
"0cym7M~W000", "service": "storage-engine", "service": "store", "path":
"/home/bpdp/.influxdbv2/engine/data"}
2022-09-16T16:14:32.367363Z    info  Compaction settings      {"log_id":
"0cym7M~W000", "service": "storage-engine", "service": "store",
"max_concurrent_compactions": 4, "throughput_bytes_per_second": 50331648,
"throughput_bytes_per_second_burst": 50331648}
2022-09-16T16:14:32.367418Z    info  Open store (start) {"log_id":
"0cym7M~W000", "service": "storage-engine", "service": "store", "op_name":
"tsdb_open", "op_event": "start"}
2022-09-16T16:14:32.367582Z    info  Open store (end)   {"log_id":
"0cym7M~W000", "service": "storage-engine", "service": "store", "op_name":
"tsdb_open", "op_event": "end", "op_elapsed": "0.169ms"}
2022-09-16T16:14:32.367663Z    info  Starting retention policy enforcement
service     {"log_id": "0cym7M~W000", "service": "retention", "check_interval":
"30m"}
2022-09-16T16:14:32.367714Z    info  Starting precreation service   {"log_id":
"0cym7M~W000", "service": "shard-precreation", "check_interval": "10m",
"advance_period": "30m"}
2022-09-16T16:14:32.369890Z    info  Starting query controller      {"log_id":
"0cym7M~W000", "service": "storage-reads", "concurrency_quota": 1024,
"initial_memory_bytes_quota_per_query": 9223372036854775807,
"memory_bytes_quota_per_query": 9223372036854775807, "max_memory_bytes": 0,
"queue_size": 1024}
2022-09-16T16:14:32.375494Z    info  Configuring InfluxQL statement executor
(zeros indicate unlimited).    {"log_id": "0cym7M~W000", "max_select_point": 0,
"max_select_series": 0, "max_select_buckets": 0}
2022-09-16T16:14:32.443622Z    info  Starting    {"log_id": "0cym7M~W000",
"service": "telemetry", "interval": "8h"}
2022-09-16T16:14:32.443927Z    info  Listening   {"log_id": "0cym7M~W000",
"service": "tcp-listener", "transport": "http", "addr": ":8086", "port": 8086}
```

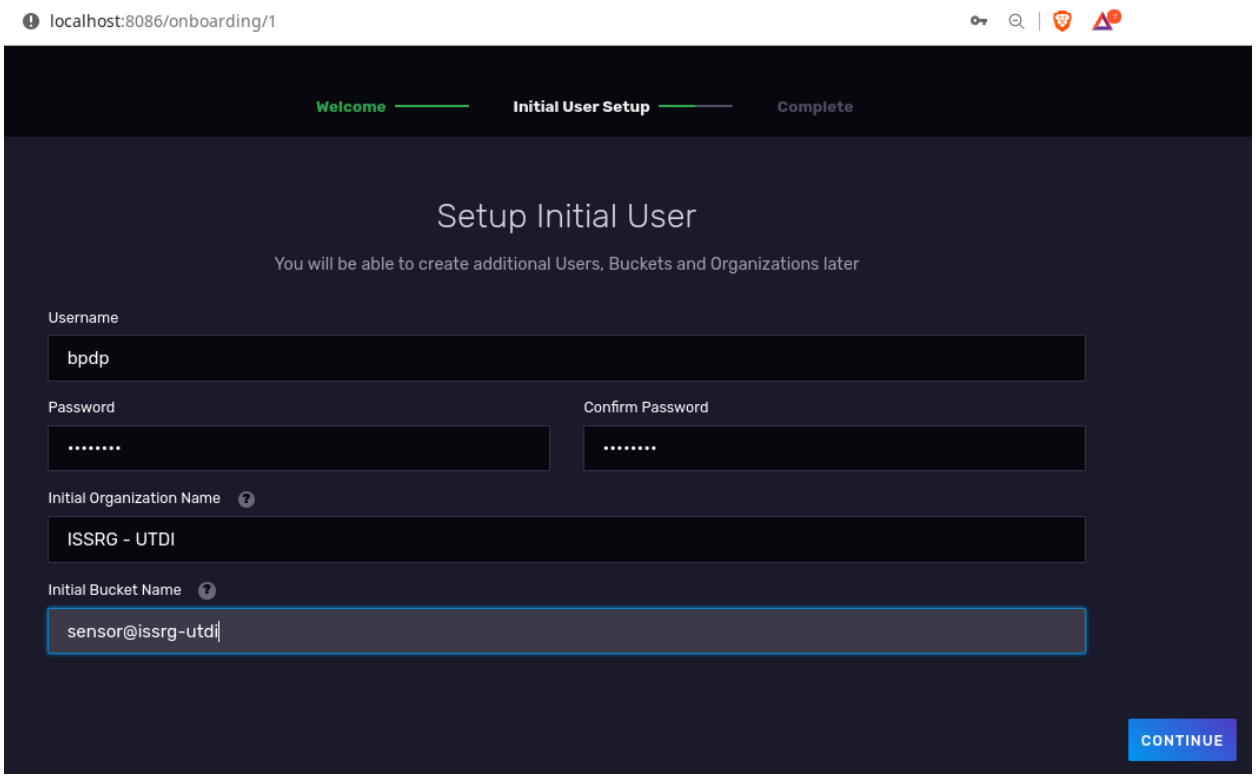
Catatan:

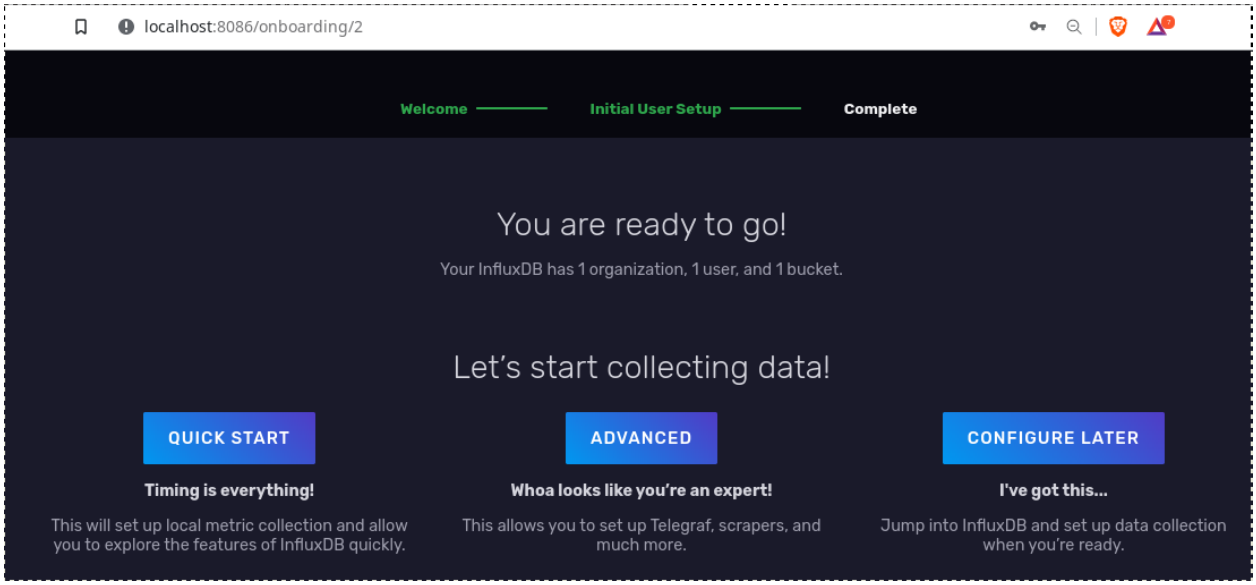
Data user, bucket, serta organisasi disimpan dalam $HOME/.influxdbv2. Jika direktori tersebut hilang, maka data user, bucket, serta organisasi hilang dan harus memulai dari baru lagi.

Setelah itu, akses dengan menggunakan Web browser ke http://localhost:8086/ untuk *getting started* - mengisikan data awal::



Setelah itu, pilih **Getting Started**, kemudian isikan:





Selesai *setup* awal InfluxDB. Pada posisi ini, kita bisa menggunakan **influx** untuk melakukan berbagai tugas administratif InfluxDB. Untuk bisa menggunakan **influx**, maka kita harus membuat konfigurasi untuk username dan password:

```
$ influx config create -n workshop-iot -u http://localhost:8086 -p
bpdp:11111111 -o "ISSRG - UTDI"
```

Contoh penggunaan CLI untuk menghapus semua data:

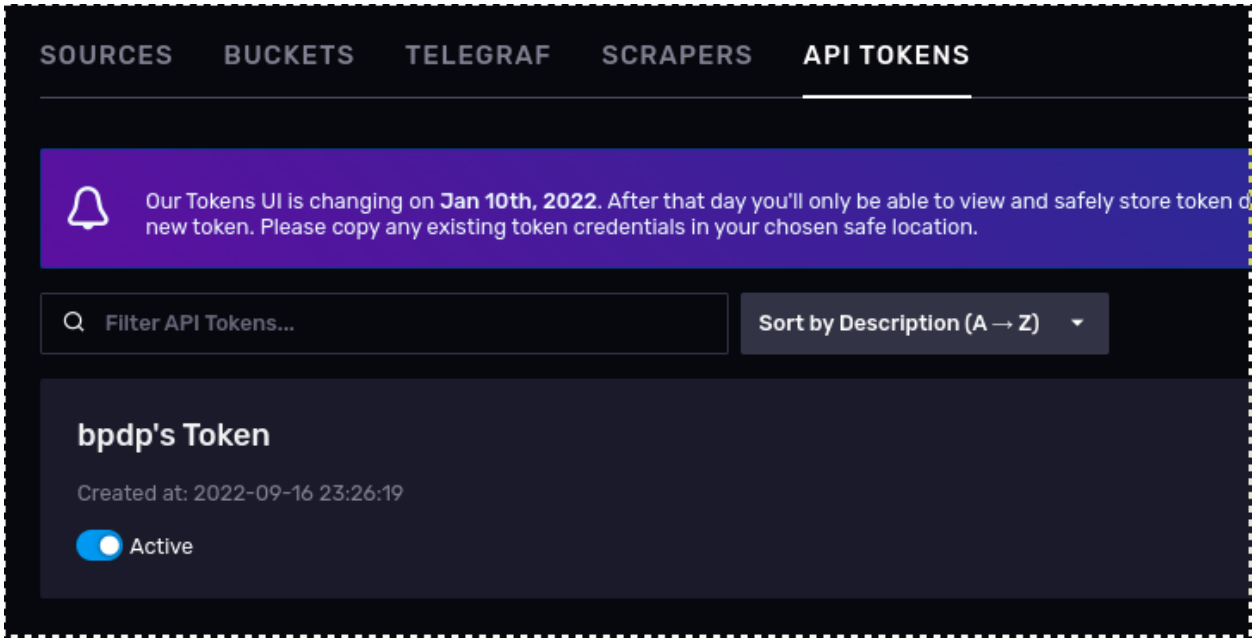
```
$ influx delete --bucket sensor@issrg-utdi --start '1970-01-01T00:00:00Z'
--stop $(date +"%Y-%m-%dT%H:%M:%SZ") --org "ISSRG - UTDI"
```

Telegraf: Konfigurasi dan Aktivasi

Konfigurasi:

```
[agent]
  ## Default data collection interval for all inputs
  interval = "1s"
  round_interval = true

  metric_batch_size = 1000

  metric_buffer_limit = 10000

  collection_jitter = "0s"

  flush_interval = "1s"
  flush_jitter = "0s"

[[inputs.mqtt_consumer]]
  servers = ["tcp://localhost:1883"]

  topics = [
    "sensors/#",
  ]
  data_format = "influx"

[[outputs.influxdb_v2]]
  urls = ["http://127.0.0.1:8086"]

  token =
"fND06E3H5SryulaKlqHmGtFvX6tYJlYwZtssjpM5b1ONDZ7VgPxdr8e3YNtRGr8V_bdZ2JpK5x154O
uoZfaNWA=="

  organization = "ISSRG - UTDI"

  bucket = "sensor@issrg-utdi"
```

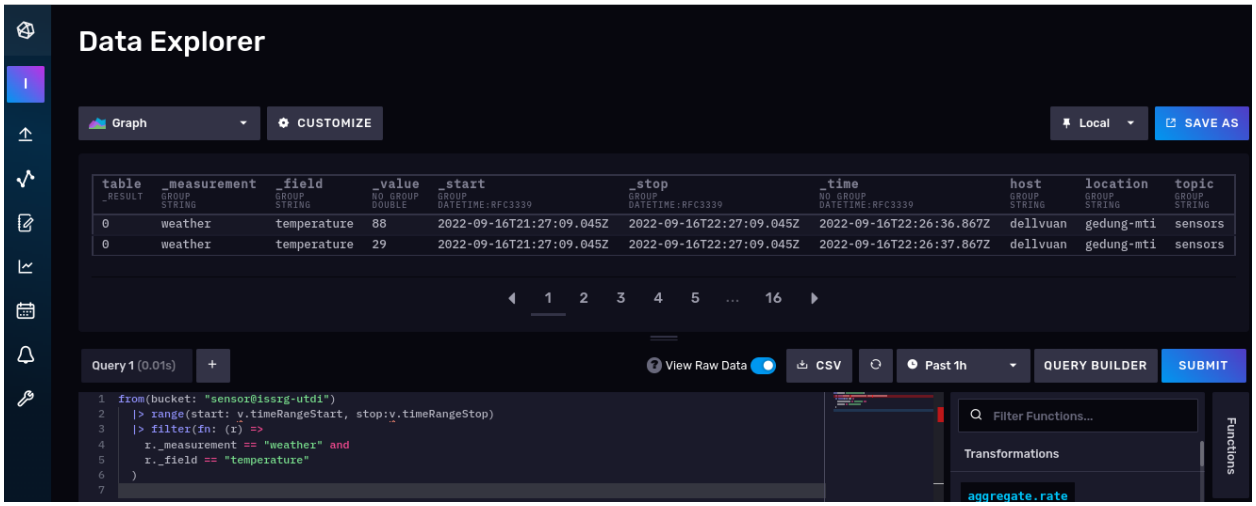
Catatan: token di-isi dengan API token sesuai user.



Setelah itu, untuk memasukkan data ke InfluxDB pada buckets sesuai yang telah ditentukan pada konfigurasi telegraf, jalankan telegraf:

```
$ telegraf
2022-09-16T22:26:36Z I! Using config file:
/home/bpdp/software/workshop-iot/telegraf/etc/telegraf/telegraf.conf
2022-09-16T22:26:36Z I! Starting Telegraf 1.24.0
2022-09-16T22:26:36Z I! Available plugins: 222 inputs, 9 aggregators, 26
processors, 20 parsers, 57 outputs
2022-09-16T22:26:36Z I! Loaded inputs: mqtt_consumer
2022-09-16T22:26:36Z I! Loaded aggregators:
2022-09-16T22:26:36Z I! Loaded processors:
2022-09-16T22:26:36Z I! Loaded outputs: influxdb_v2
2022-09-16T22:26:36Z I! Tags enabled: host=dellvuan
2022-09-16T22:26:36Z I! [agent] Config: Interval:1s, Quiet:false,
Hostname:"dellvuan", Flush Interval:1s
2022-09-16T22:26:36Z I! [inputs.mqtt_consumer] Connected [tcp://localhost:1883]
```

Untuk melihat bahwa data sudah masuk, maka kita bisa melakukan query menggunakan query editor pada buckets:



Coba perintah berikut:

Untuk menampilkan weather dan temperature

```
from(bucket: "sensor@issrg-utdi")
  |> range(start: v.timeRangeStart, stop:v.timeRangeStop)
  |> filter(fn: (r) =>
    r._measurement == "weather" and
    r._field == "temperature"
  )
```

Untuk menampilkan data yang masuk selama 15 menit terakhir

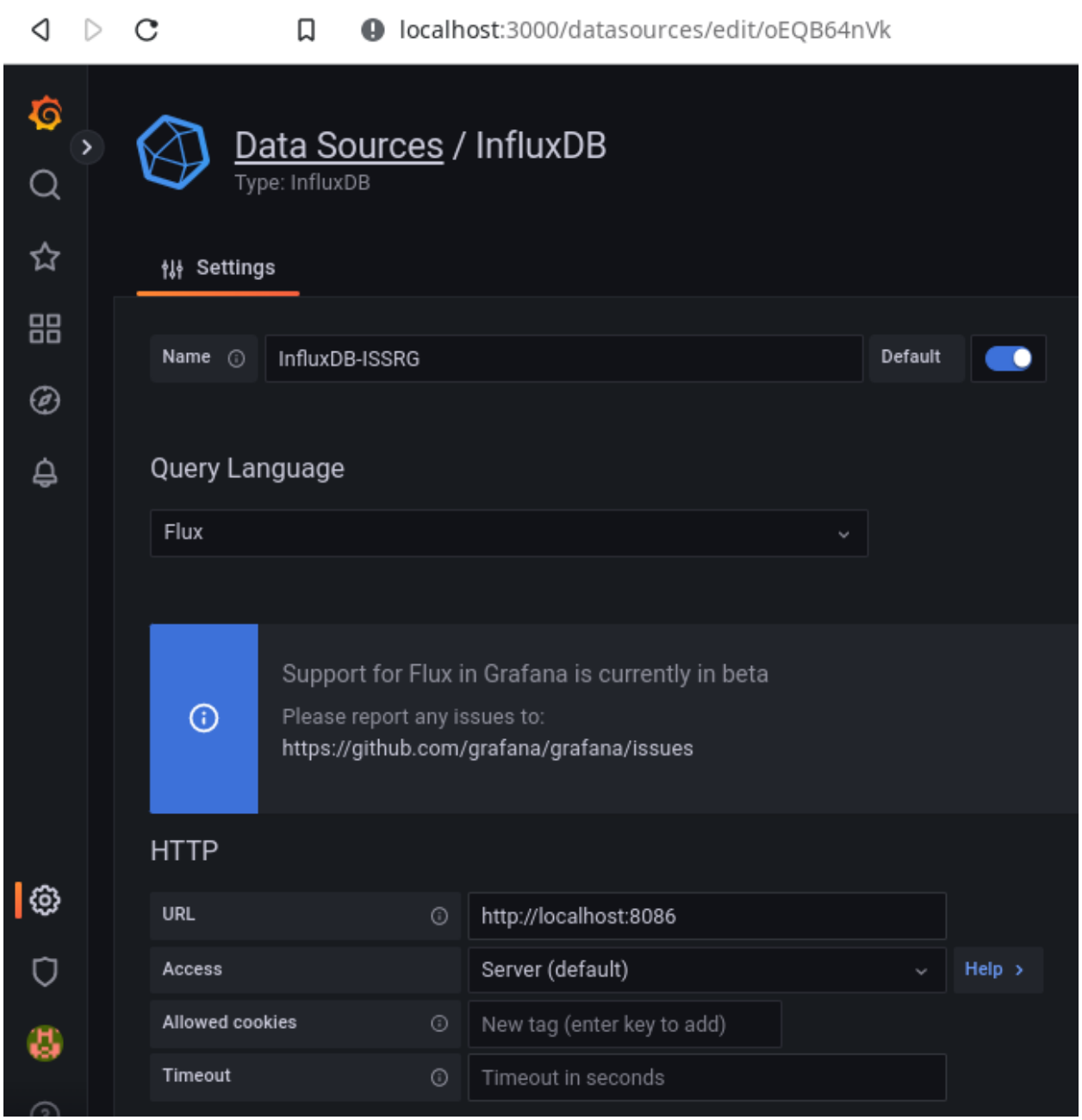
```
from(bucket:"sensor@issrg-utdi")
    |> range(start: -15m)
```

Visualisasi Menggunakan Grafana

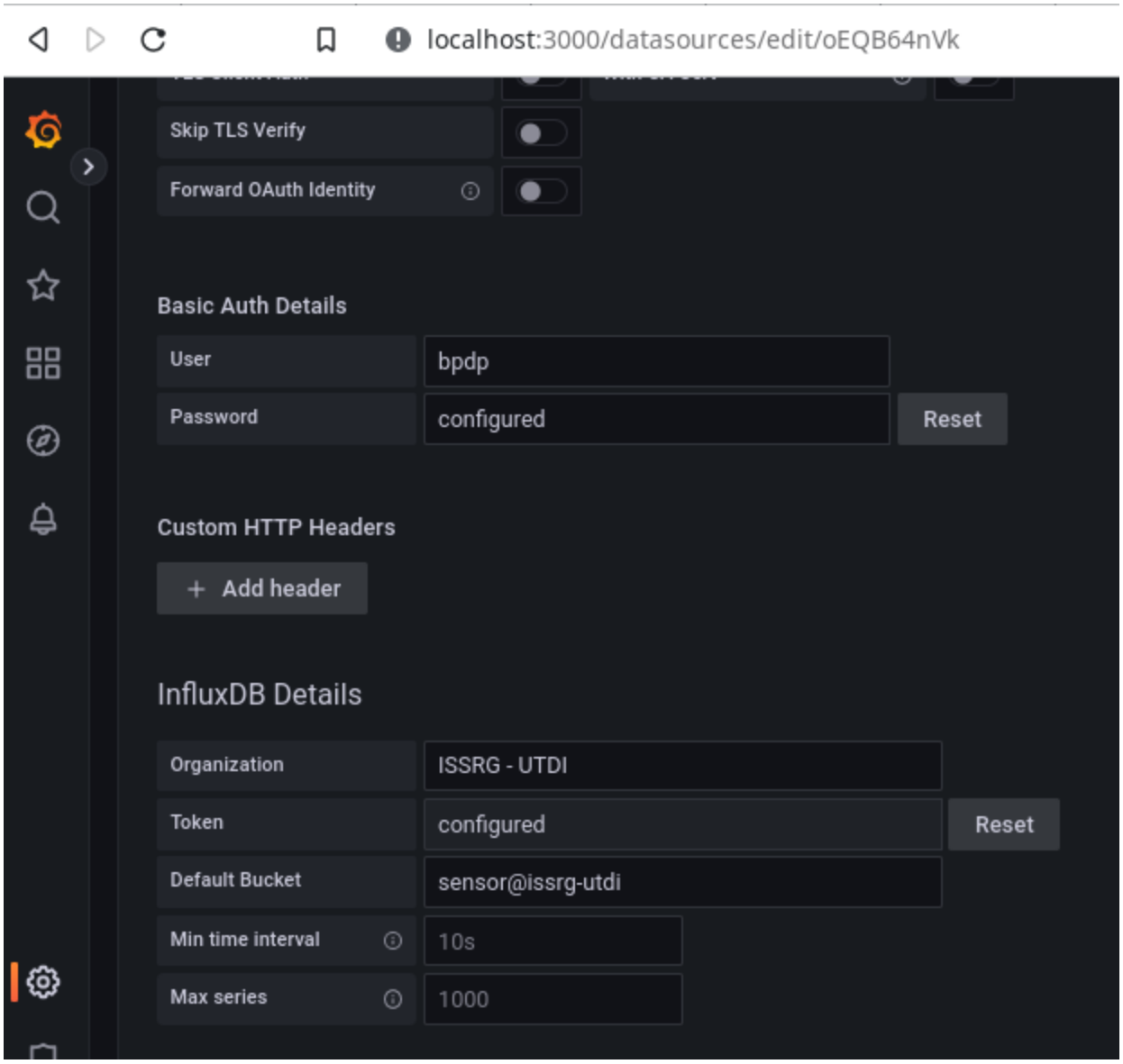
Setelah install **grafana**, jalankan **grafana-server** berikut:

```
$ grafana-server -homepath ~/software/workshop-iot/grafana-oss/
```

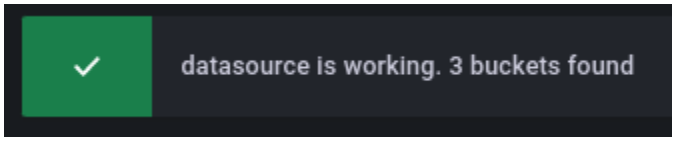
Akses ke Web UI dengan http://localhost:3000. Setelah itu, pilih **Data Sources**.



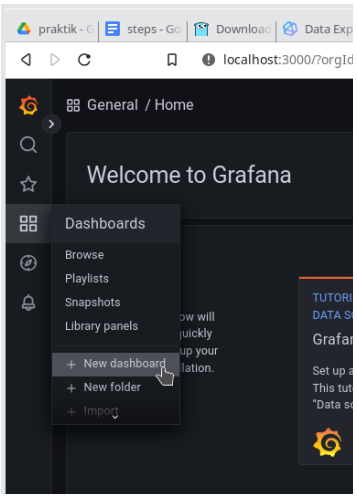
Pilih **flux** untuk *Query Language*. Isikan URL sesuai dengan URL untuk InfluxDB. Pada bagian bawah, isikan *Basic Authentication* dengan user dan password dari InfluxDB yang pertama kali dibuat. Isikan juga *Organization*, *Token,* serta *Bucket* sesuai dengan yang ada pada InfluxDB.



Setelah itu **Save and Test**. Jika berhasil, maka akan muncul:

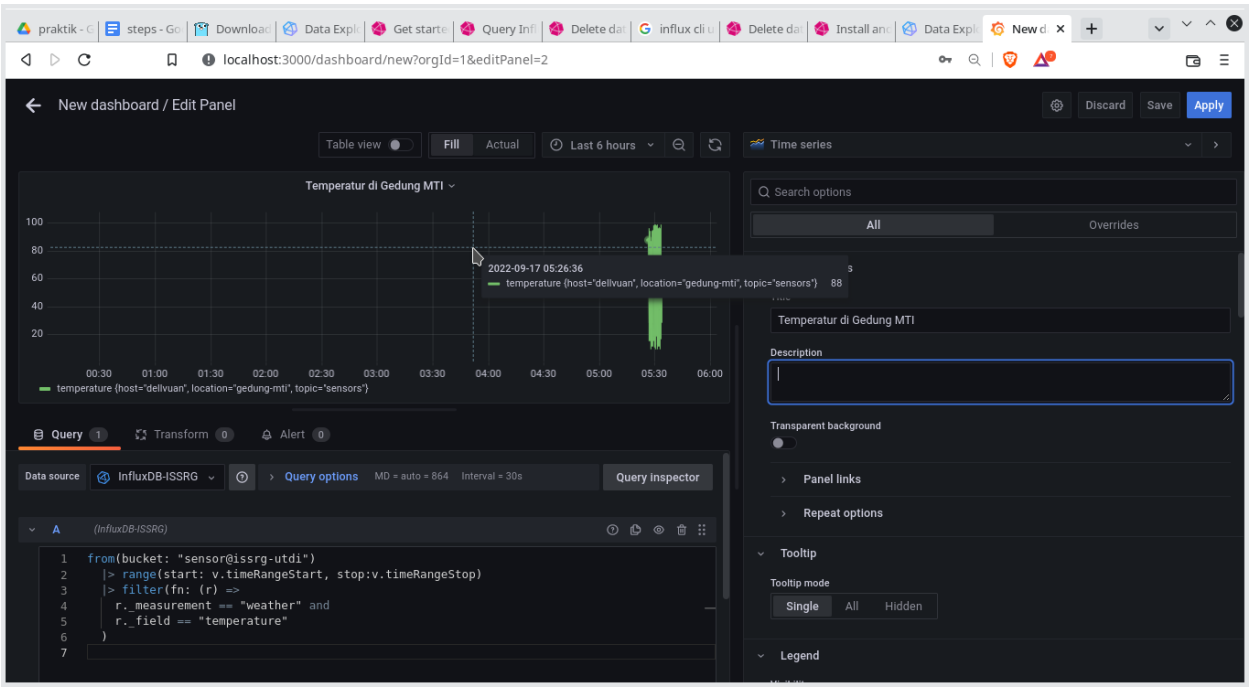


Untuk membuat dashboard, pilih:

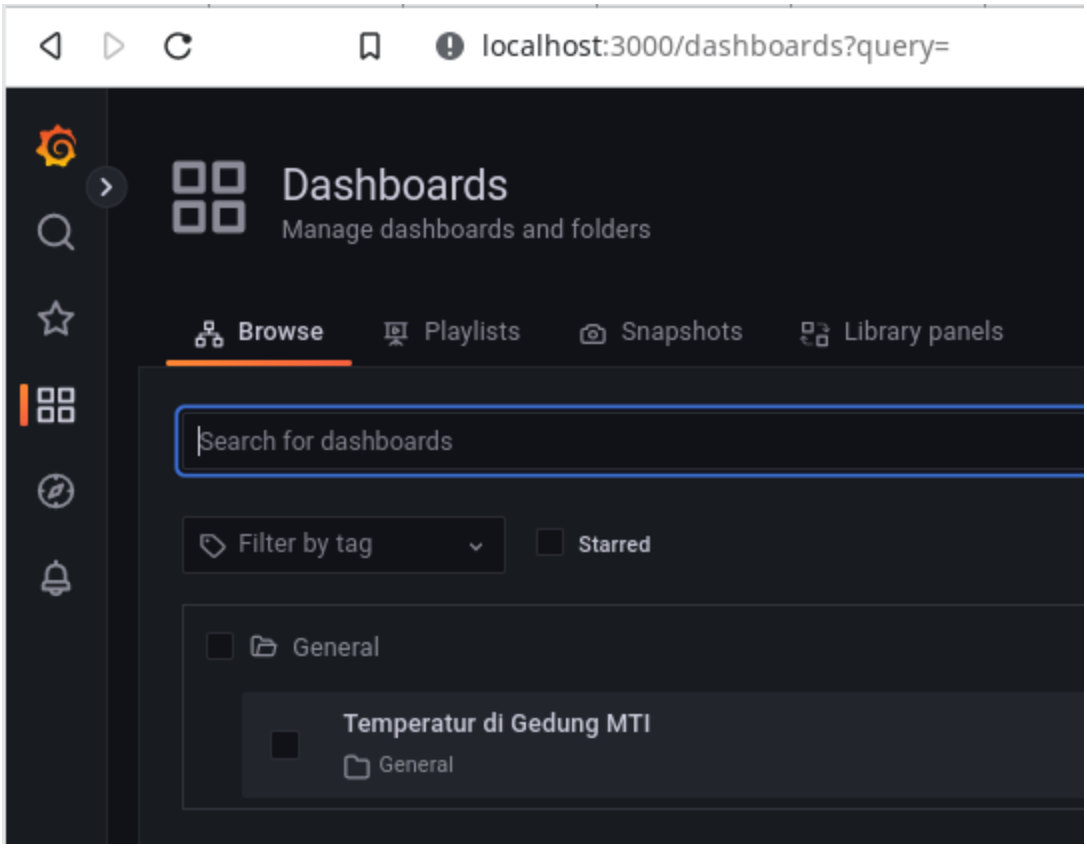


Setelah itu, pilih **Add a New Panel**. Isikan query data pada bagian bawah:

```
from(bucket: "sensor@issrg-utdi")
  |> range(start: v.timeRangeStart, stop:v.timeRangeStop)
  |> filter(fn: (r) =>
    r._measurement == "weather" and
    r._field == "temperature"
  )
```



Setelah disimpan, Dashboard tersebut bisa kita lihat. Jika data dinamis dan berupa streaming, maka dashboard akan berubah sesuai dengan kondisi data.



Dashboard tersebut bisa kita klik untuk menampilkan kondisi berdasarkan sensor yang telah kita pasang: