

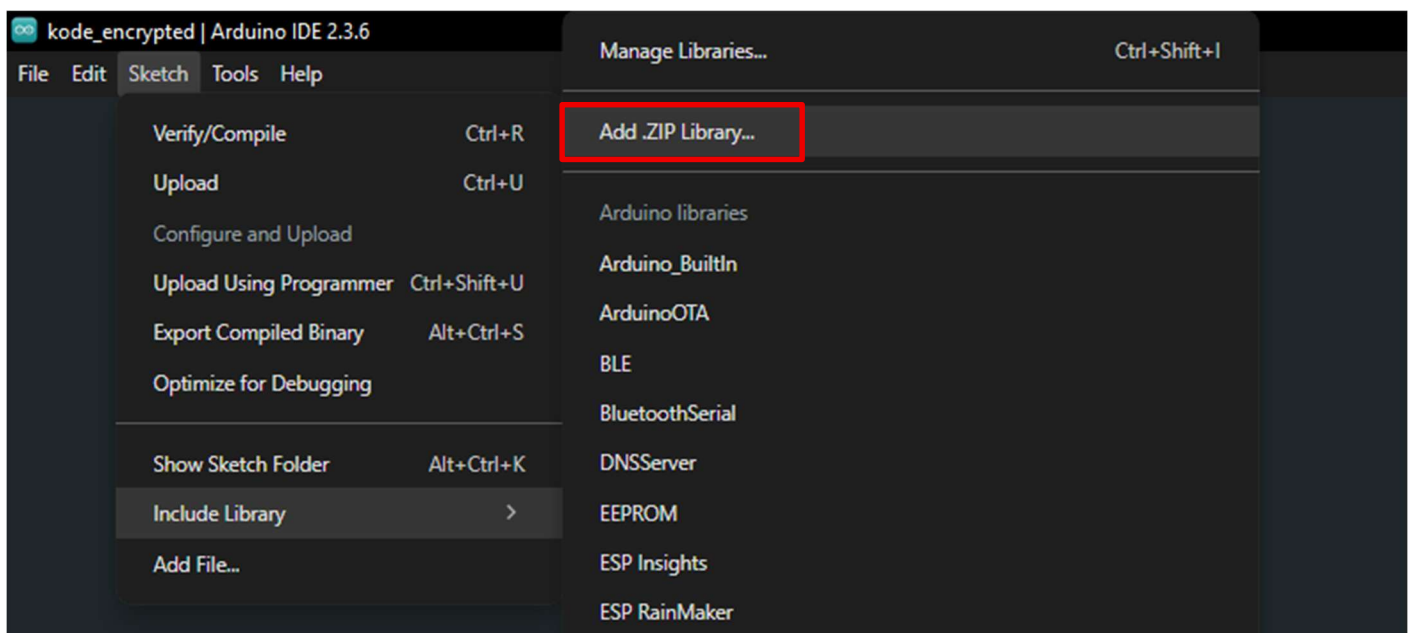
TUTORIAL Enkripsi Dengan Algoritma Kriptografi Ringan: ASCON-128

Perkembangan *Internet of Things* (IoT) telah mendorong kebutuhan akan sistem keamanan yang efisien, terutama karena perangkat IoT umumnya memiliki **keterbatasan dalam daya komputasi, memori, dan konsumsi energi**. Kriptografi konvensional seperti AES sering kali terlalu berat untuk diterapkan secara optimal di perangkat-perangkat ini. Oleh karena itu, dibutuhkan algoritma **kriptografi ringan** yang tetap mampu menjamin **kerahasiaan, integritas, dan autentikasi data**.

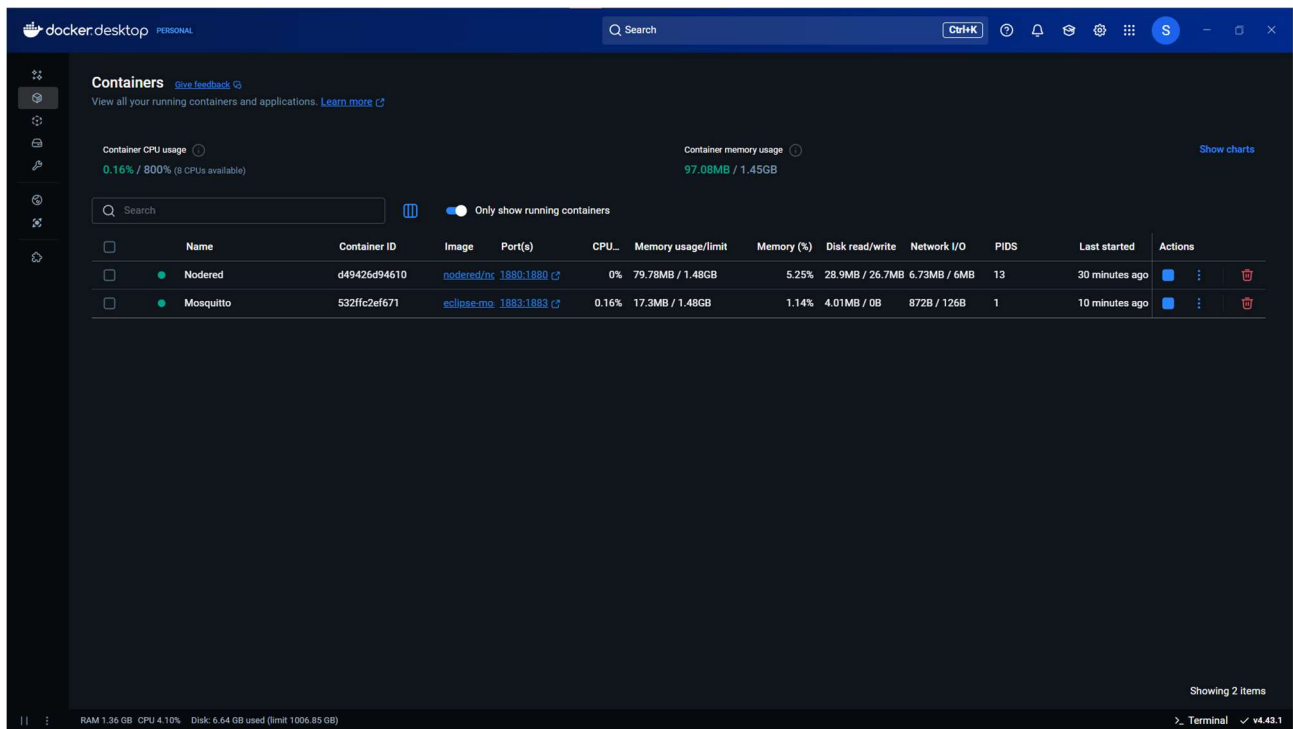
ASCON-128 yang dirancang khusus untuk **efisiensi** dan **keamanan** pada perangkat IoT dengan **sumber daya terbatas**. Dipilih sebagai salah satu algoritma standar oleh *National Institute of Standards and Technology* (NIST) untuk **kriptografi ringan**, **ASCON-128** menawarkan perlindungan data dengan konsumsi sumber daya yang minimal, menjadikannya ideal untuk implementasi di ekosistem IoT yang luas dan beragam.

Langkah-langkah:

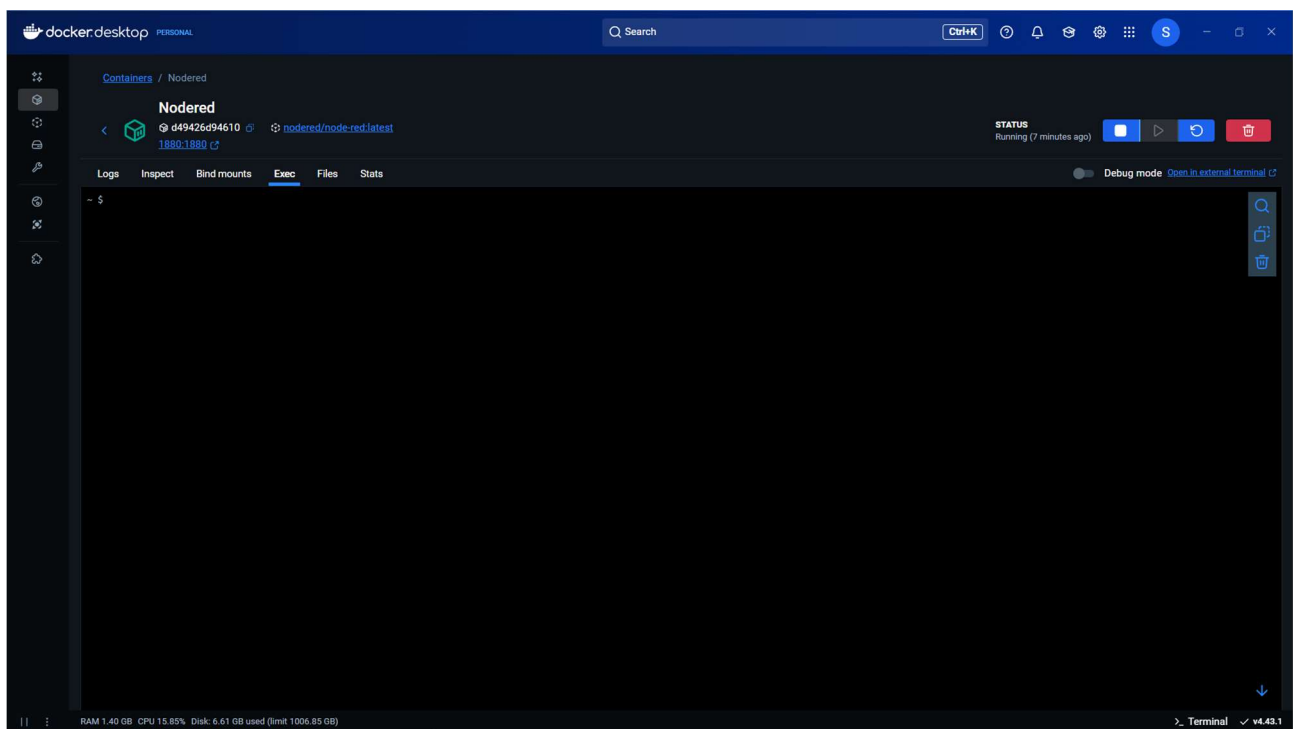
1. Pertama *download* **kode node** dan **kode gateway** pada github. Sesuaikan **UUID** pada **kode node** dan sesuaikan **kode gateway** dari (**nama wifi, password wifi, ip mqtt, UUID dan Key ASCON**)
[MATKUL_KEAMANAN-JARINGAN/ENKRIPSI/Kode_at_main · mhmdnvn18/MATKUL_KEAMANAN-JARINGAN](https://github.com/mhmdnvn18/MATKUL_KEAMANAN-JARINGAN/ENKRIPSI/Kode_at_main)
2. Lalu *download* dan *install* kedua *library* ASCON-128 ke Arduino IDE pada github
https://github.com/mhmdnvn18/MATKUL_KEAMANAN-JARINGAN/tree/main/ENKRIPSI/Library



3. Buka Docker Dekstop, *Run* Mosquitto dan *Run* Node-RED nya

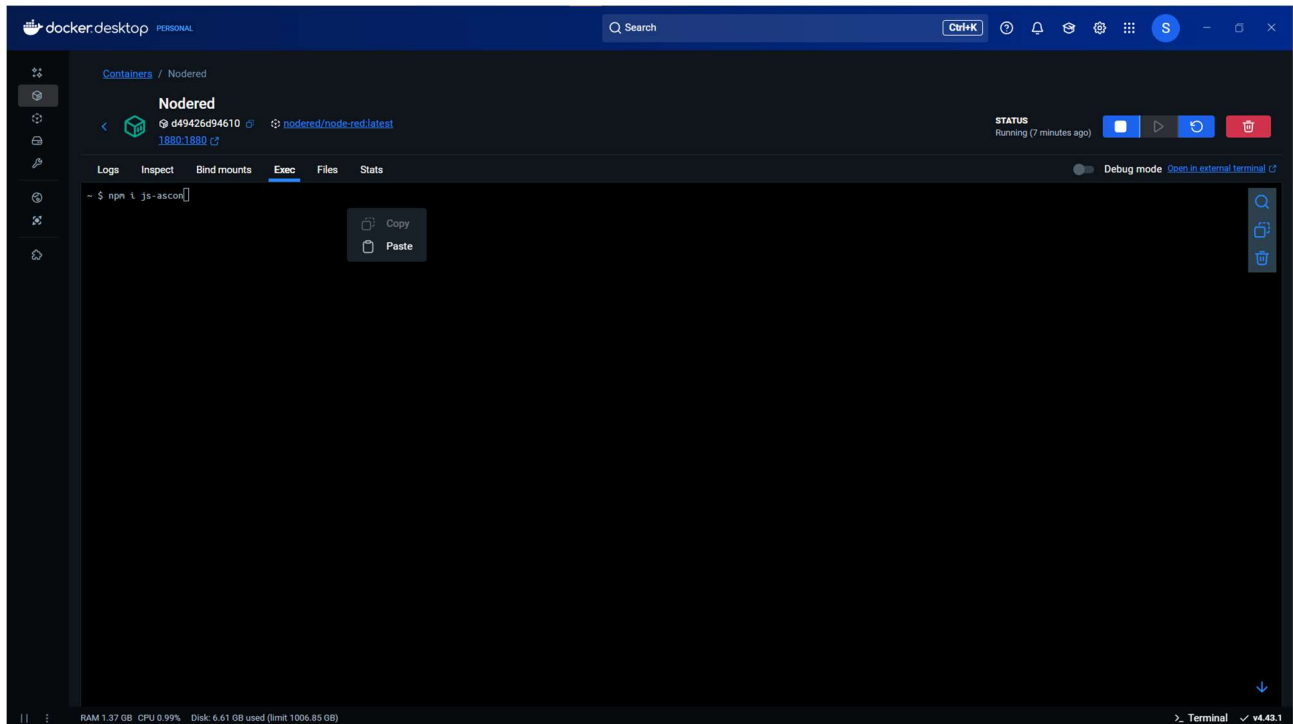


4. Masuk ke Node-Red, dengan klik Nodered, lalu masuk ke exec

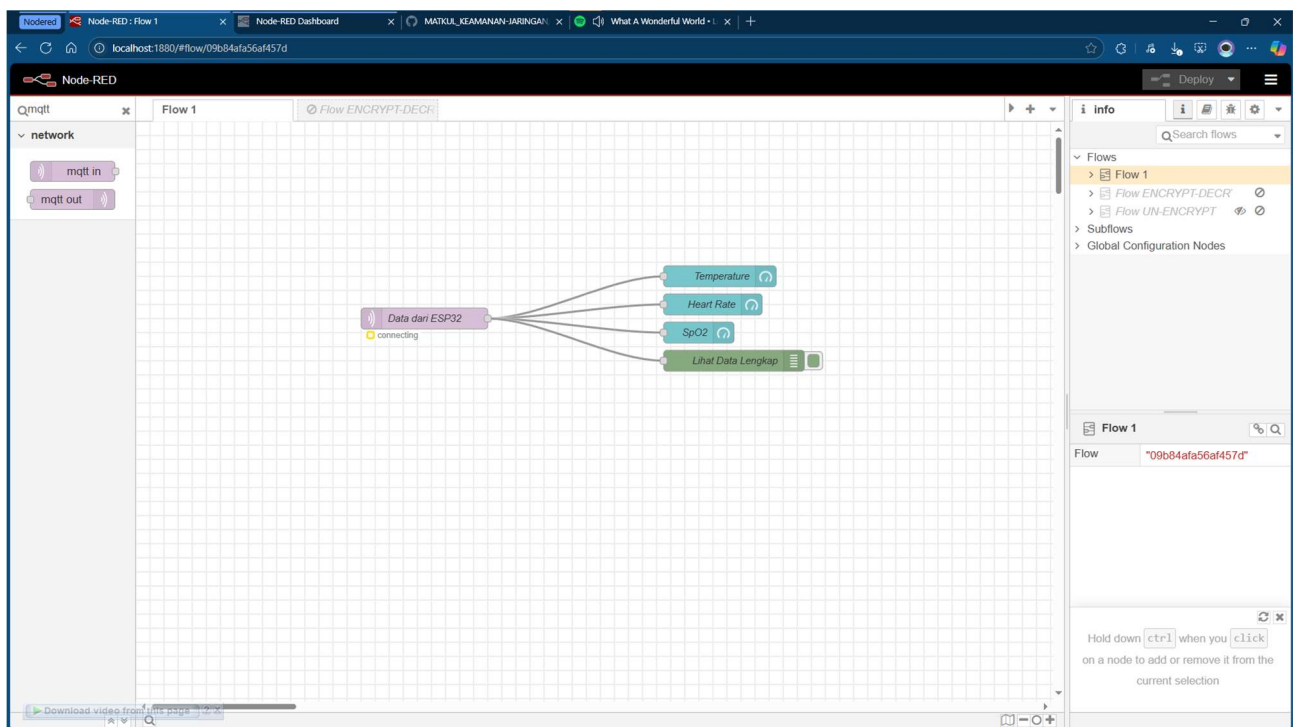


5. Paste command ***npm i js-ascon*** dengan cara klik kanan > *paste*. Lalu Enter.

Command ini digunakan untuk menginstal *library* ASCON ke Node-Red

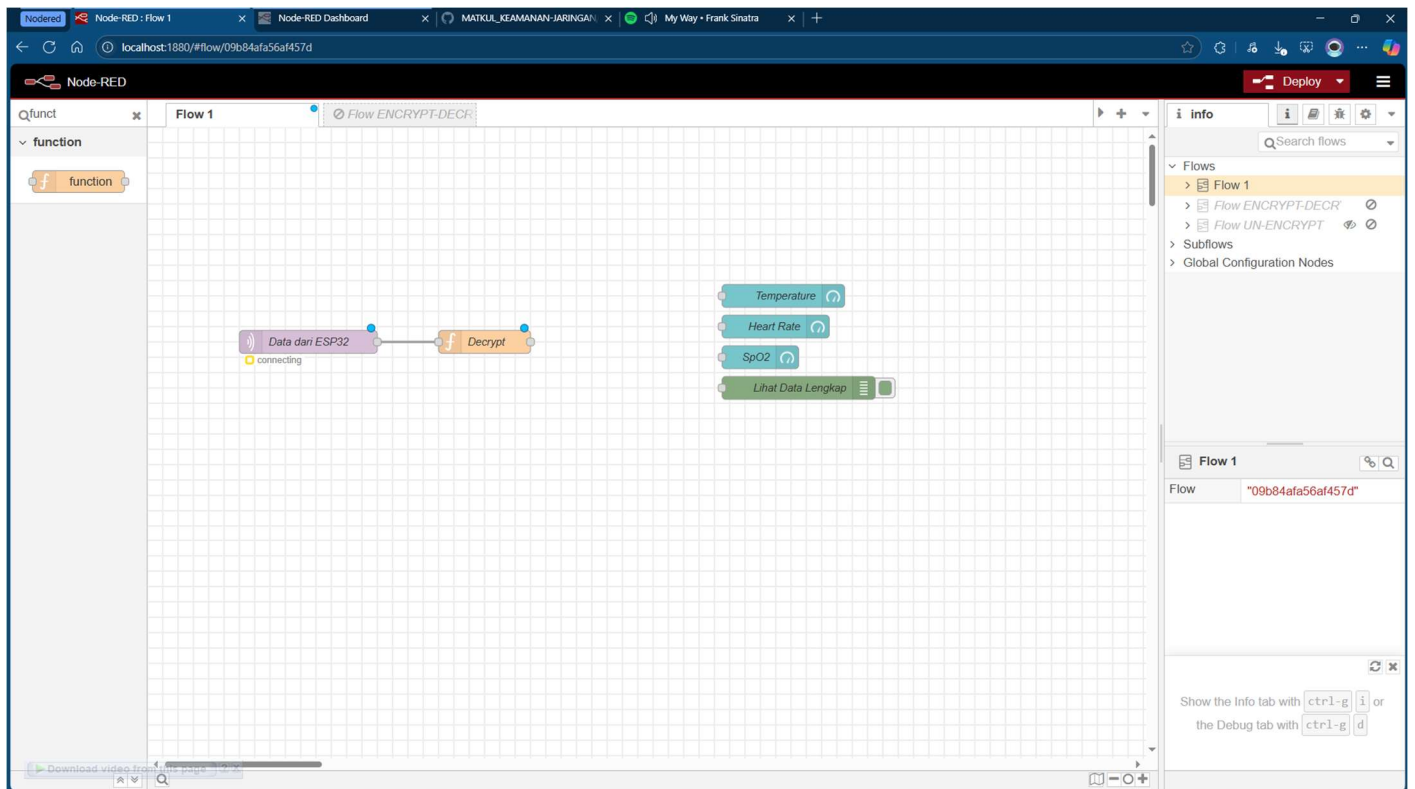


6. Buka *browser* lalu jalankan nodered, ketik **localhost:1880** Maka akan tertampil *flow* yang sudah dibuat pada pertemuan sebelumnya.

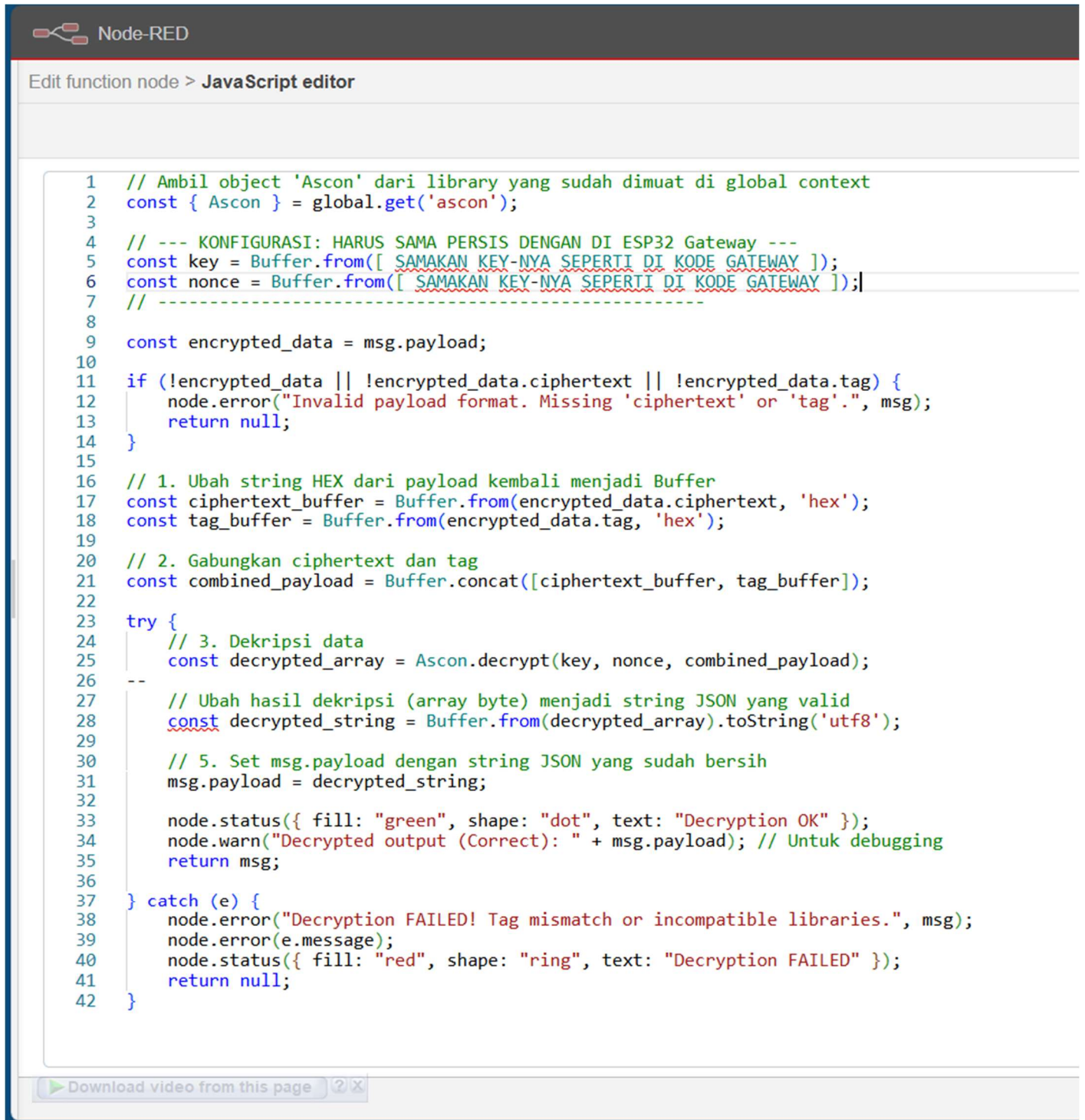


7. Lalu tambahkan *flow function*,

Flow ini digunakan untuk sebagai dekripsi ASCON-128 dari sisi *backend* sebelum ditampilkan pada ui



9. Double klik **function**, lalu isi *name* dengan *Decrypt* dan lalu isi pada **On Message** dengan *command* seperti gambar di bawah. Lalu **DONE**

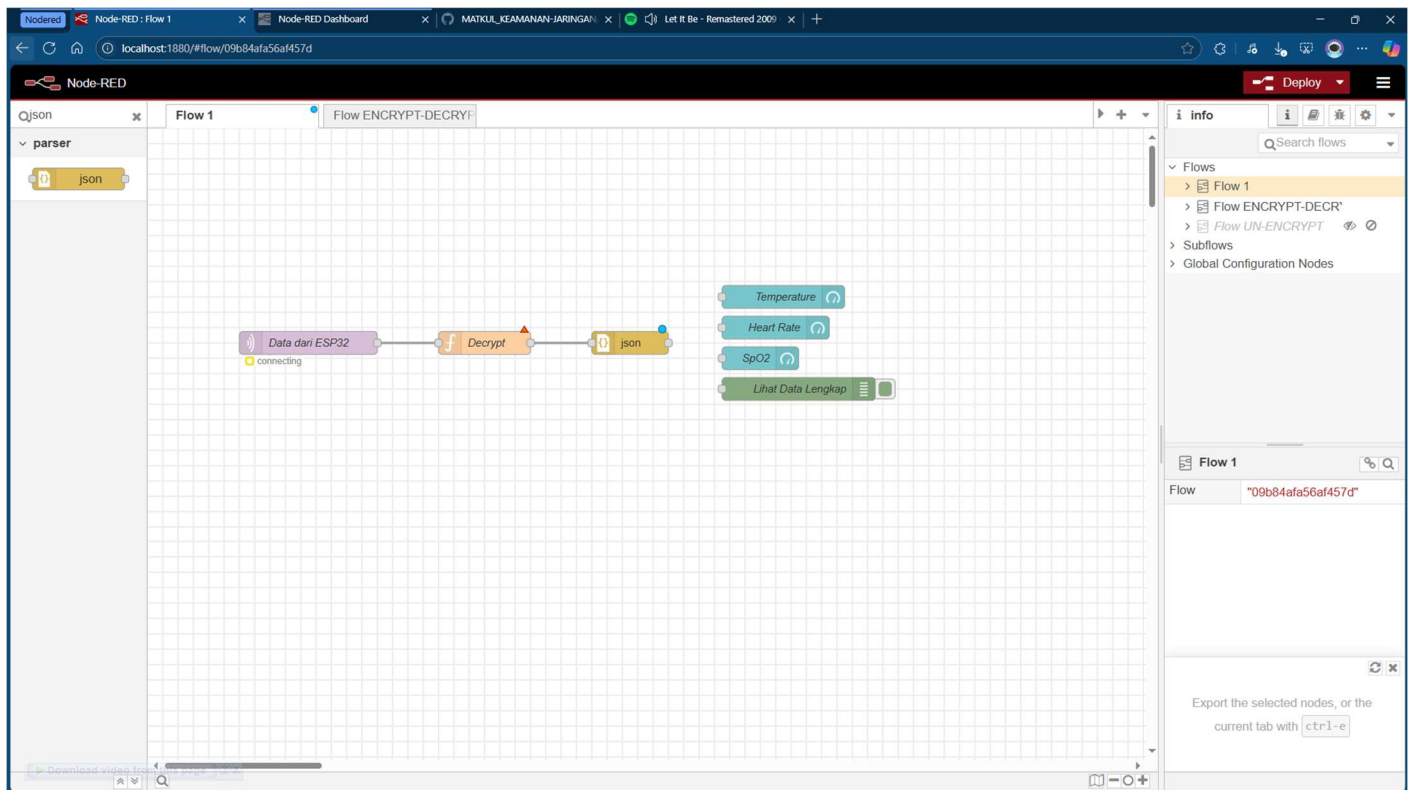


The screenshot shows the Node-RED JavaScript editor with a function node named 'Decrypt'. The code implements a decryption process using the 'ascon' library. It first checks if the payload has 'ciphertext' and 'tag' properties. If not, it returns an error. Then, it converts the ciphertext and tag from hex strings to buffers, concatenates them, and decrypts the combined payload using the 'ascon' library. The result is converted back to a UTF-8 string and set as the message payload. Status and warning messages are used for debugging.

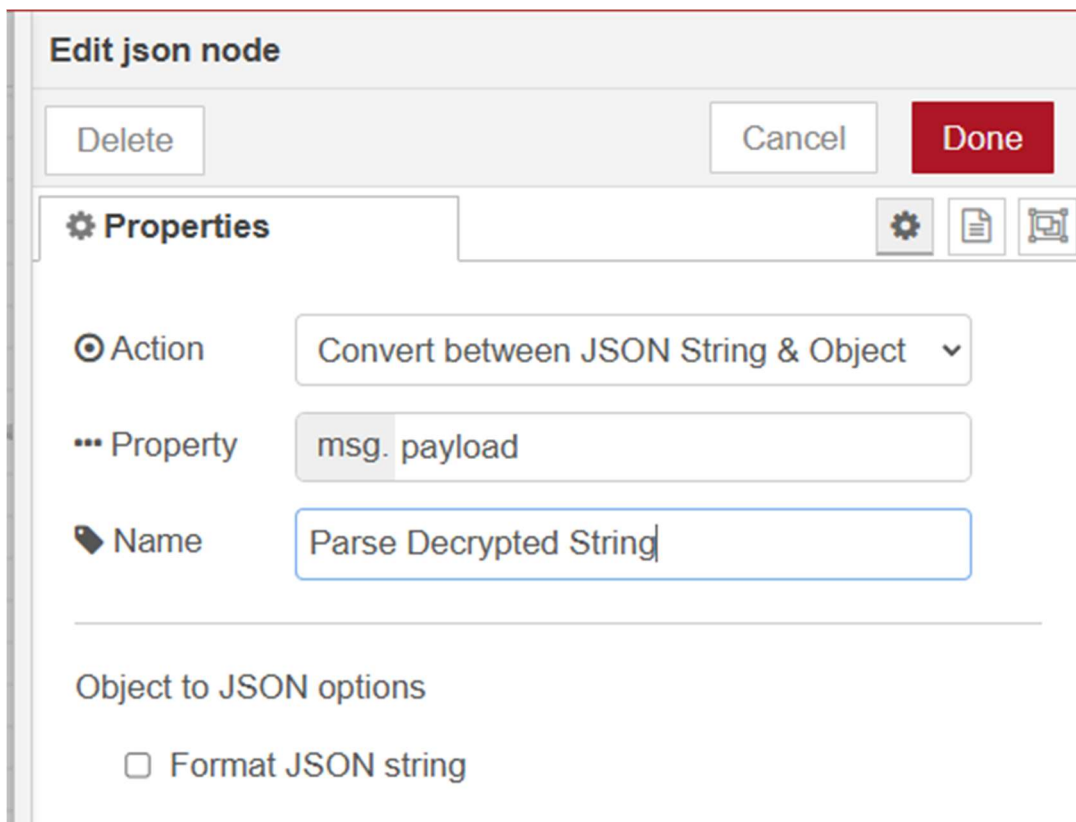
```
1 // Ambil object 'Ascon' dari library yang sudah dimuat di global context
2 const { Ascon } = global.get('ascon');
3
4 // --- KONFIGURASI: HARUS SAMA PERSIS DENGAN DI ESP32 Gateway ---
5 const key = Buffer.from([ SAMAKAN KEY-NYA SEPERTI DI KODE GATEWAY ]);
6 const nonce = Buffer.from([ SAMAKAN KEY-NYA SEPERTI DI KODE GATEWAY ]);
7 // -----
8
9 const encrypted_data = msg.payload;
10
11 if (!encrypted_data || !encrypted_data.ciphertext || !encrypted_data.tag) {
12   node.error("Invalid payload format. Missing 'ciphertext' or 'tag'.", msg);
13   return null;
14 }
15
16 // 1. Ubah string HEX dari payload kembali menjadi Buffer
17 const ciphertext_buffer = Buffer.from(encrypted_data.ciphertext, 'hex');
18 const tag_buffer = Buffer.from(encrypted_data.tag, 'hex');
19
20 // 2. Gabungkan ciphertext dan tag
21 const combined_payload = Buffer.concat([ciphertext_buffer, tag_buffer]);
22
23 try {
24   // 3. Dekripsi data
25   const decrypted_array = Ascon.decrypt(key, nonce, combined_payload);
26   --
27   // Ubah hasil dekripsi (array byte) menjadi string JSON yang valid
28   const decrypted_string = Buffer.from(decrypted_array).toString('utf8');
29
30   // 5. Set msg.payload dengan string JSON yang sudah bersih
31   msg.payload = decrypted_string;
32
33   node.status({ fill: "green", shape: "dot", text: "Decryption OK" });
34   node.warn("Decrypted output (Correct): " + msg.payload); // Untuk debugging
35   return msg;
36 } catch (e) {
37   node.error("Decryption FAILED! Tag mismatch or incompatible libraries.", msg);
38   node.error(e.message);
39   node.status({ fill: "red", shape: "ring", text: "Decryption FAILED" });
40   return null;
41 }
42 }
```

Download video from this page

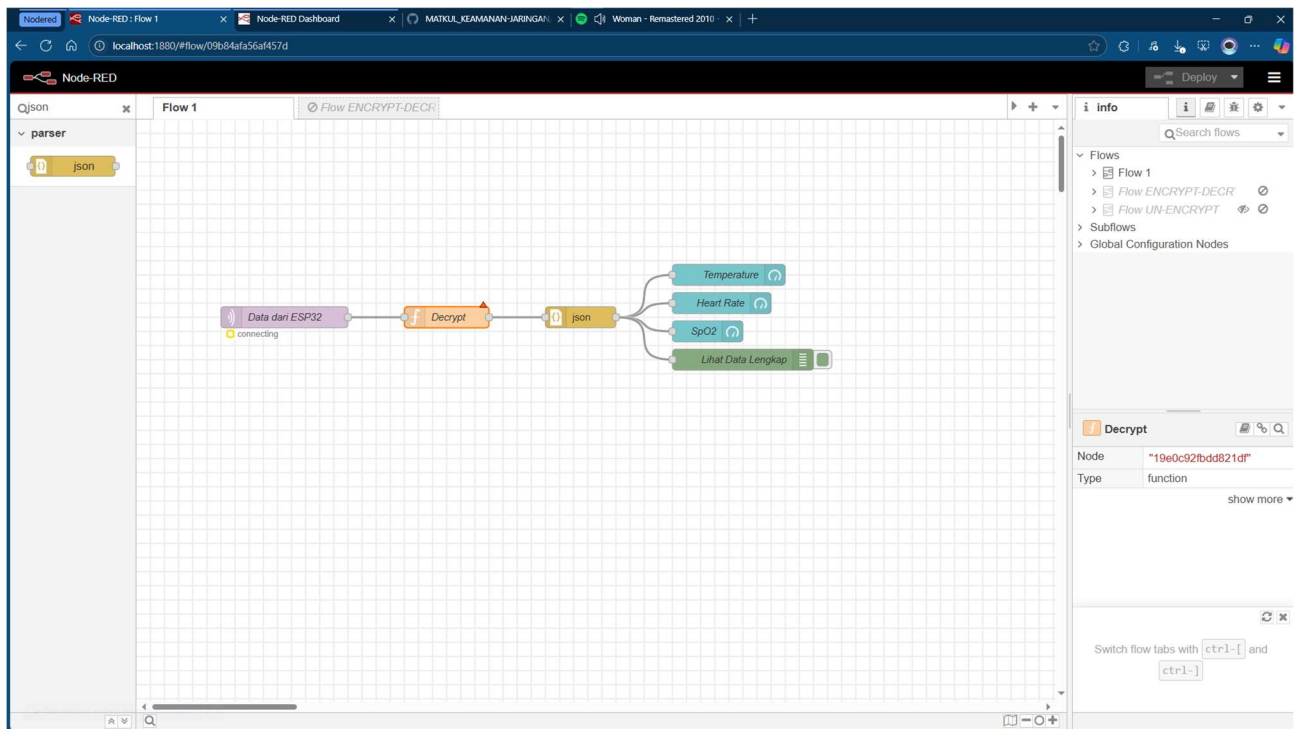
10. Lalu tambahkan *flow JSON*,



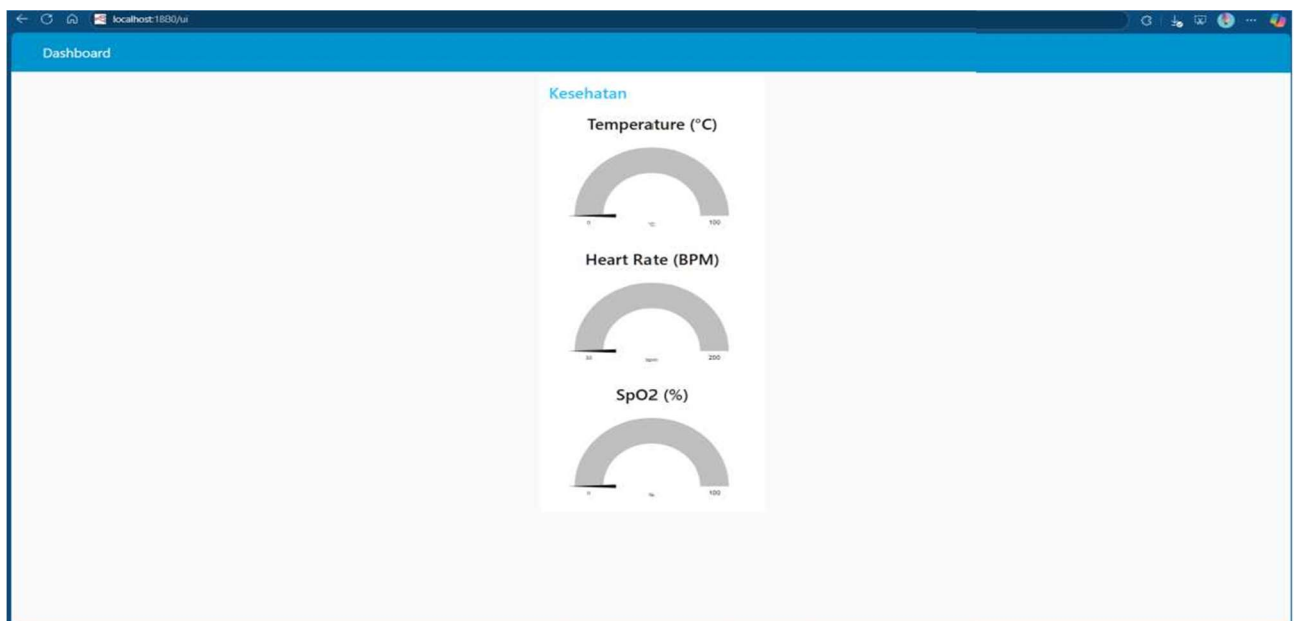
11. Double klik **json**, isi parameter seperti gambar di bawah. Lalu **DONE**.



12. Sambungkan semua *flow*-nya. Lalu *DEPLOY*.



13. Buat tab baru lalu ketikkan <http://localhost:1880/ui>



14. Uji keseluruhan sistemnya.